NETWORK TRANSFORMATIONS AND SOME APPLICATIONS

Yue Pui Cheong

NAVAL POSTGRADUATE SCHOOL
Monterey, California

THESIS

NETWORK TRANSFORMATIONS
AND SOME APPLICATIONS

by

Yue Pui Cheong

December 1975

Thesis Advisor: Gordon H. Bradley

Unclassified

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER	2. GOVT ACCESSION NO.	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) Network Transformations and Some Applications		5. TYPE OF REPORT & PERIOD COVERED Master's Thesis; December 1975 6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s) Yue Pui Cheong		8. CONTRACT OR GRANT NUMBER(s)
9. PERFORMING ORGANIZATION NAME AND ADDRESS Naval Postgraduate School Monterey, California 93940		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS
11. CONTROLLING OFFICE NAME AND ADDRESS Naval Postgraduate School Monterey, California 93940		12. REPORT DATE December 1975 13. NUMBER OF PAGES
14. MONITORING AGENCY NAME & ADDRESS(if different from Controlling Office) Naval Postgraduate School Monterey, California		15. SECURITY CLASS. (of this report) Unclassified 15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

Approved for public release; distribution unlimited.

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

Capacitated networks
multicommodity flow problem
GNET solution code
network transformations

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

The growing number of large scale applications of network models and the availability of very fast solution codes make it attractive to formulate problems as networks whenever such models are adequate for the purpose. In this thesis, conceptualization of, and notation used to express these models is based on the interpretation of physical flows of commodity through a network structure of nodes and arcs. As an aid to modelling, and to allow codes of varying

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 68 IS OBSOLETE
S/N 0102-014-6601 |
(Page 1)

Unclassified

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

specificity to be used, nine well-known Transformations are catalogued here for easy reference.

Two recent results for special cases of the multicommodity flow problem are re-derived and in the case of (1) below, is significantly extended: (1) The case with all capacitated arcs in the network structure incident with one common node. (2) The case of transportation structure with two sinks (or two sources). Using the network approach, these are shown to have equivalent network formulations.

Lastly, a Transformation which uncapacitates a network is implemented in various ways into a contemporary solution code named GNET.

Network Transformations and Some Applications

by

Yue Pui Cheong
Major, Singapore Armed Forces
B.S., Queen's University of Belfast, 1966

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN OPERATIONS RESEARCH

from the

ABSTRACT

The growing number of large scale applications of network models and the availability of very fast solution codes make it attractive to formulate problems as networks whenever such models are adequate for the purpose. In this thesis, conceptualization of, and notation used to express these models is based on the interpretation of physical flows of commodity through a network structure of nodes and arcs. As an aid to modelling, and to allow codes of varying specificity to be used, nine well-known Transformations are catalogued here for easy reference.

Two recent results for special cases of the multicommodity flow problem are re-derived and in the case of (1) below, is significantly extended: (1) The case with all capacitated arcs in the network structure incident with one common node. (2) The case of a transportation structure with two sinks (or two sources). Using the network approach, these are shown to have equivalent network formulations.

Lastly, a Transformation which uncapacitates a network is implemented in various ways into a contemporary solution code named GNET.

TABLE OF CONTENTS

I. INTRODUCTION---------- 9

A. NETWORK MODELS---------- 0

B. VIEWPOINTS----------11

C. THE MULTICOMMODITY FLOW PROBLEM----------13

D. IMPORTANCE OF NETWORK TRANSFORMATIONS AND CONTRIBUTION OF THE THESIS----------13

E. SOME IMPORTANT PROPERTIES OF NETWORK MODELS-------16

II. NETWORK TRANSFORMATIONS----------18

A. USES OF TRANSFORMATIONS----------18

B. EQUIVALENCE OF NETWORKS AND INVERSE TRANSFORMATIONS----------19

C. EXISTENCE OF EQUIVALENT NETWORK MODELS----------20

D. THE TRANSFORMATIONS----------22

1. Transformation I----------22

2. Transformation II----------24

3. Transformation III----------27

4. Transformation IV----------28

5. Transformation V----------30

6. Transformation VI---------- 33

7. Transformation VII---------- 35

8. Transformation VIII---------- 36

9. Transformation IX---------- 37

III. TWO SPECIAL CASES OF THE MULTICOMMODITY FLOW PROBLEM---------- 39

A. GENERAL DISCUSSION---------- 39

B. RESULT I--THE "BUSY NODE" PROBLEM---------- 41

1. Constructive Derivation of the Result---------43

2. Complex Joint Capacities----------49

3. Extensions of the Result----------51

C. RESULT II--THE TWO-SINK (OR SOURCE) TRANSPORTATION PROBLEM----------56

IV. UNCAPACITATING A TRANSSHIPMENT NETWORK--COMPUTER IMPLEMENTATION----------63

A. GENERAL CONSIDERATIONS----------63

B. ASPECTS OF THE IMPLEMENTATION----------64

C. DESCRIPTION OF THE CODE----------65

1. Arc Data Structure----------66

2. SIMPLEX Data Structure----------67

3. Preprocessor LNET----------70

D. THE CHOSEN TRANSFORMATION----------70

E. THE CHOSEN PROBLEMS----------71

F. BASIC DATA CHANGES----------72

G. INSERTION OF DATA CHANGES INTO THE PROGRAM--THREE CASES----------74

1. Within LNET----------75

2. Pre-hanging----------75

3. Hang-on-the-fly----------78

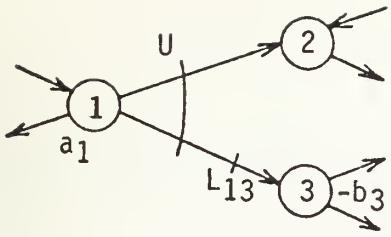
H. RESULTS AND DISCUSSION----------79

LIST OF REFERENCES----------81

INITIAL DISTRIBUTION LIST----------82

LIST OF SYMBOLS AND NOTATION

N,A	The set of nodes, and directed arcs, respectively, which define the structure of a network.
x_{ij}	Flow along arc (i,j) from node i to node j. x_{ij} is taken as ≥ 0, with flows entering and leaving a node having opposite signs.
U_{ij}	Upper bound on flow along arc (i,j).
U	Upper bound on sum of flows along a specified set of jointly capacitated arcs.
L_{ij}, L	Signifies lower bounds, as U_{ij} and U signify upper bounds. If omitted, L is understood to be zero.
T_i	Supply/demand at node i. Used when it is not necessary to distinguish between a supply ($T > 0$) and demand ($T < 0$).
a_i	A supply of a units available at node i.
$-b_j$	A demand of b units at node j, ($b_i > 0$).
c_{ij}	The unit cost of transportation along arc (i,j).
m	Depending on context, signifies: a. Total number of sources in the standard transportation problem, or b. Total number of sources which are connected by permissible arcs to a particular node in a transshipment node.
n	As for m, applied to sinks instead of sources.
Superscript number	Signifies the commodity corresponding to that number in a multicommodity problem, e.g., x^2_{ij} is the flow of commodity 2 along arc (i,j).



An example showing part of a transshipment network. Note that the short arrows incident to the nodes indicate existence of other arcs and nodes not drawn--only the illustrated subnetwork is of interest. Node 1 is a source with supply a_1 and node 3 is a sink with demand $-b_3$. Arcs (1,2) and (1,3) have a joint upper capacity of U units, and (1,3) has a lower bound of L_{13} units. The curved lines intersect arcs to which the bounds apply.

Third subscript

Signifies a member of a set of multiple arcs between two nodes, e.g., x_{ijk} is the flow along the k^{th} of a set of arcs from node i to node j.

I. INTRODUCTION

A. NETWORK MODELS

A large and growing variety of problems in areas such as the transportation of goods, design of communications and pipeline systems, assignment of men to jobs, bid evaluation and production planning are adequately and accurately described by network models.

These problems usually occur in the context of transporting a single homogeneous commodity from a set of points called "source nodes" which generate supplies of the commodity (e.g., factories, refineries) to another set of points called "demand nodes" with demands for the commodity (e.g., retail stores, end users), through a transportation system or "network" which consists of a set of notes (including the sources and sinks) connected by a set of directional routes or "arcs" (e.g., roads, pipelines). Each arc connecting a pair of nodes has a cost of transportation per unit of commodity, and perhaps an upper bound or "capacity" on the total flow of commodity along it. The objective is to find a minimum-cost flow pattern through the network which satisfies all demands.

By network models is meant the special class of minimum cost (and maximum flow as a particular case) linear programming (L.P.) models of which the most general is the single-commodity capacitated transshipment model, usually formulated as follows:

Minimize $\sum_i \sum_j c_{ij} x_{ij}$

subject to $\sum_j x_{ij} - \sum_k x_{ki} = T_i$ for each node i, this equation expressing the algebraic relation between incoming and outgoing flow at the node,

and $L_{ij} \leq x_{ij} \leq U_{ij}$ for each arc (i,j).

Non-negativity of the x_{ij} and equality of total supply to total demand is necessary in the above equality formulation of the model. Positive lower bounds, L_{ij} on x_{ij} are easily dealt with by simple arithmetical transformations to yield variables with zero-valued lower bounds.

Other models in this class are more specific cases of the above formulation, and include the capacitated and uncapacitated transportation model and the personnel assignment model. All these models are well-known and widely described, for example, in Ref. [9].

Although the models are conceptually and algebraically simple, the real-life problems they describe are typically huge, and it has been found necessary to find economical means of solving problems with, for example, 50,000 nodes and 500,000 arcs.

B. VIEWPOINTS FOR NETWORK MODELS

The notation used in expressing network models is important because it influences the development and implementation of

network algorithms, and is relevant to the ease with which certain theoretical results are conceived and proven.

One way of viewing a network model is as a straightforward L.P. With slacks introduced to convert the inequality capacity constraints to equalities, the constraint expressions in the above formulation may be economically written in the usual matrix form MX = T. L.P. solution codes accept input and manipulate data in matrix and vector form, and theoretical results arrived at using this approach involve operations borrowed from linear and matrix algebra.

Another viewpoint, which may be called the network approach, is derived from the actual physical nature of many problems--that of finding an optimal pattern of flows x_{ij} through a network structure consisting of directed, capacitated (perhaps) arcs (i,j) which connect nodes at which are demands/supplies T_i. This viewpoint deals with arcs rather than vectors, and interprets the L.P. constraint equations as expressions of flow conservation at a node.

The network approach is more natural than the use of the notation in the above formulation of a network model, since:

a. In practical network problems, it is rarely the case that every node pair is joined by an arc. There is also the possibility of multiple arcs between a node pair. It is thus more appropriate to formulate the problem in terms of the set A of arcs which are actually present in the network structure:

Minimize $\sum_{(i,j)\in A} c_{ij} x_{ij}$

subject to $\sum_{j|(i,j)\in A} x_{ij} - \sum_{k|(k,i)\in A} x_{ki} = T_i$, for each node i,

and $L_{ij} \leq x_{ij} \leq U_{ij}$, for each $(i,j)\in A$.

b. Data input and manipulation in all efficient network codes is in arc form. The storage and manipulation of large sparse matrices implicit in L.P. codes is by comparison inefficient and cumbersome.

In view of the growing size of real-life applications, a principal practical result of the network approach has been the development of solution codes which are much faster than L.P. codes for problems of equal size. Although they follow precisely the SIMPLEX algorithm as do L.P. codes, network-based codes accept, store and manipulate data in a manner which is based on the interpretation of a basic set of variables as a tree which spans the nodes of the problem (Koopmans [6]). "Tree" is a term from graph theory which describes a network which has only one (unique) path between any two nodes, disregarding arc orientation.

The differences between the two viewpoints lie in visual conceptualization, terminology, and solution implementation, rather than in any difference in mathematical foundation. The solution algorithm is the same in both, and as an example of their commonality in mathematical basis (which lies in linear algebra), it is noted that the network transformations

introduced below may be viewed simply as linear operations on the matrix M and vector T. In many respects, the two viewpoints are complementary in rigor and ease of intuitive appreciation. The physical interpretation of the problem in the network approach does make it particularly acceptable to non-analysts.

C. THE MULTICOMMODITY FLOW PROBLEM

The general multicommodity flow problem is a minimum-cost transshipment problem in which a number of distinguishable commodities flow along the capacitated directed arcs of a network structure in accordance with separate demands and supplies for each commodity at the nodes, and an arc's capacity applies to the sum of flows of the commodities along it. There has been recent interest in finding special cases which may be transformed into network models in order to take advantage of fast network codes, the more so because a problem with r commodities has approximately r times the number of variables of a single commodity problem with the same network structure.

D. IMPORTANCE OF NETWORK TRANSFORMATIONS AND CONTRIBUTION OF THIS THESIS

Because of the size of many practical applications, and the availability of contemporary network-based codes which can be faster than L.P. codes by a factor of 100 or more, it is obviously advantageous to formulate a problem as a network model wherever possible.

It will be noted that capacitated network models have capacities which are imposed only on individual arcs.

There are many problems which have capacities imposed on nodes, and on sets of arcs (a "joint capacity" being an upper bound on the sum of flows through a specified set of arcs). Such problems cannot directly take advantage of network-based codes, and must be re-formulated or transformed into an equivalent capacitated or uncapacitated network model (there are, however, certain types of joint capacity described later which do not permit this transformation).

It is also important to be able to transform network models amongst themselves (e.g., transshipment to transportation, capacitated to uncapacitated) to suit the specificity of a particular solution code, since not all network-based codes can solve the general capacitated transshipment problem.

For the above purposes, there have been a series of transformations proposed and used over the past 20 years or so to deal with networks which are jointly capacitated in a number of ways. The main work of this thesis lies in:

1. Cataloging these transformations into a single reference as an aid to modelling.

2. Developing a unified approach to the many transformations by showing that they may be constructed by selective sequential application of a very few basic transformations.

3. Re-derivation of two recent results (and in one case, significant extension) involving special cases of the multicommodity flow problem, using the network transformation approach. The two cases of interest here are:

a. The two-commodity transportation problem where all capacitated arcs are incident with a single node (Rebman [7]). Here, the original result is significantly extended to the r-commodity transshipment case, in which the permissible arcs need not be identical for all commodities, and an arc's capacity need not apply to all commodities flowing along it.

b. The multicommodity transportation problem with either two sources or two sinks.

This application is an illustration of the utility of network transformations in constructive proofs of the existence of an equivalent network model for a given flow problem.

4. The implementation in various ways of a transformation designed to uncapacitate a transshipment network, using an existing network-based code GNET, which is capable of solving the capacitated transshipment problem. The speed of solution with the transformation implemented is examined to see if the saving in required memory space for a projected uncapacitated version of GNET would be worthwhile as an offset against any reduction in speed which may be caused by implementation of the transformation.

It is emphasized that the transformations derived here are not new, and most of them have their origins buried deep in the "folklore" of this field. This thesis will attempt to give credit where it is known, but an apology in advance to originators not mentioned herein is not out of order.

E. SOME IMPORTANT PROPERTIES OF NETWORK MODELS

The constraint matrix M in the formulation $MX = T$ of a network model has the property of total unimodularity, for which a necessary and sufficient condition is that every square nonsingular submatrix of M has a determinant of value -1 or 1. There other kinds of models which have this property, but all practical totally unimodular problems have so far been shown to have network model formulations.

From the L.P. viewpoint, the transformations derived in this thesis are viewed as reformulations of the original problems in such a way that the matrix M of the transformed problem becomes (or remains) totally unimodular.

A major consequence of total unimodularity is that every basic solution (and an optimal solution) is integer valued, given integer T in $MX = T$. This is of importance in many real problems for which only integer answers make sense. It is also true that if the optimal solution to an L.P. problem is unique and fractional, then the problem cannot be formulated as a network model.

There are other classes of L.P. problems which are interpretable as finding optimal flows through a network structure but which are not totally unimodular and thus cannot be formulated as a network model as defined here, except in special cases. Examples are the network flow problem with gains, and the general multicommodity problem, which can in fact be thought of as a single-commodity problem with an expanded network structure made up of disjoint sub-networks (one for each commodity)

identical to the original structure, with a certain type of joint capacity which does not permit a network model formulation. Such flow problems are not termed "network models" in the sense used herein.

A network structure is said to be bipartite if its nodes can be exhaustively divided into two disjoint subsets such that every arc has one incident node in one subset and the other incident node in the other subset, and all arcs (directed) are oriented in the same direction from one of the subsets to the other. The structures of the transportation and assignment problems are bipartite.

In the terminology used here, nodes through which the commodity is permitted to flow en route to other notes are called transshipment nodes. A node at which total incoming commodity is constrained to equal total outgoing commodity (i.e., at which demand = supply = 0) is called a "pure transshipment" node. Sources or sinks which are not also transshipment nodes are called "pure source" or "pure sink" nodes, otherwise "demand transshipment" or "supply transshipment" is used. When it is unnecessary to make distinctions, the word "pure" will be omitted.

II. NETWORK TRANSFORMATIONS

Of the following Transformations, the first three may be regarded as basic in the sense that they are used in the construction of almost all the succeeding (more complex) transformations.

Transformations are named and referred to by Roman numerals, and are spelled with a capital "T" to distinguish them from normal usage of the word, in this and subsequent sections.

A. USES OF TRANSFORMATIONS

These Transformations have been constructed with one or more of the following uses in mind:

a. To simplify a complex capacitated structure (e.g., transforming a group of arcs with a joint capacity into arcs with individual capacities).

b. To uncapacitate a capacitated network.

c. To transform a transshipment network to a transportation network.

d. A combination of some or all of a, b, and c, above.

e. To simplify a network, in the sense of reducing the number of arcs and/or nodes.

The broader purposes of using transformations lie in:

1. Transforming a problem into a form suitable for use with the particular type of network-based code available. The specificity of application of a code may vary from the uncapacitated transportation problem to the general transshipment problem.

For problems of equal size (i.e., with the same number of arcs and nodes), a more specific code should be faster than a general code. On the other hand, transforming a network into a form which can be handled by a more specific code almost invariably carries the penalty of increasing the size of the network due to the addition of artificial arcs and/or nodes. Transformations which reduce the size of a network carry the reverse penalty of increasing the generality of the code required, either because they introduce more complex capacitating mechanisms or because they transform a transportation structure to a transshipment structure. There is therefore an inverse relationship between the size and the generality of the many forms a problem may take.

2. Use in the constructive derivation of results, as is done later in this thesis.

B. EQUIVALENCE OF NETWORKS AND INVERSE TRANSFORMATIONS

The idea of "equivalence" in transforming one network into another is defined in terms of being able to obtain the optimal solution to the original network by performing post-optimal arithmetic operations on the optimal solution (objective function and/or flows) to the transformed network. It is necessary for equivalence of two networks that their objective functions differ only by a known additive or multiplicative factor.

Equivalence also implies "invertibility" of a transformation to obtain the original network. However, given only the transformed network, the applicability of the inverse transformation is difficult to see in most cases. In fact, it is

rare to find a network with the precise features necessary for application of many inverses (e.g., zero-cost arcs).

C. EXISTENCE OF EQUIVALENT NETWORK MODELS

In order for a network flow problem with joint capacities to have an equivalent network formulation, it is necessary for its joint capacities to satisfy the following well-known conditions (e.g., Dantzig [3]):

Condition I. At any node, any two joint capacities imposed on its incident arcs either have no arcs in common, or the arcs of one joint capacity are a subset of the other joint capacity (i.e., any two joint capacities at a node apply to arc sets which are either disjoint or such that one arc set is a subset ("nested") of the other).

Condition II. A joint capacity is not imposed on arcs any two or more of which are completely node disjoint (i.e., a joint capacity should be applied to a set of arcs which has at least one common incident node).

Condition III. A joint capacity at a node is not imposed on a set of arcs which has some member(s) oriented in the opposite direction to some other member(s), with respect to the node. The exception is when all the arcs are incident between the same node pair. In this case, only one of the arcs will have a non-zero flow in any basis. Thus, the problem can be reformulated with each arc individually capacitated with the original joint capacity.

Figure 1 illustrates violation of Conditions I and II, and Figure 2 illustrates violation of Condition III.

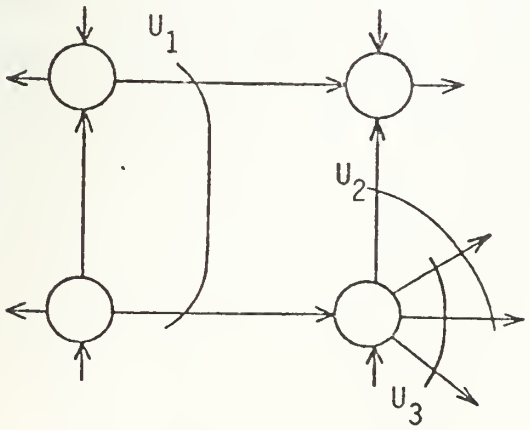


Fig. 1

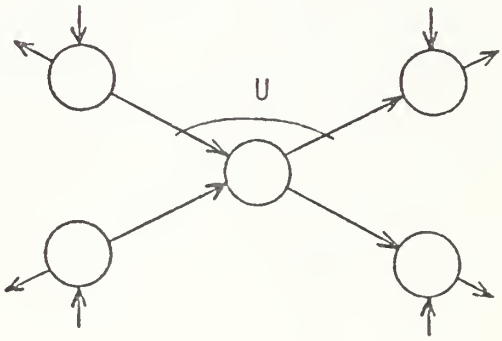


Fig. 2

When a joint capacity appears visually to violate Condition II, it is sometimes profitable to look at both ends of the arcs in question. For example, a first glance at Fig. 3 indicates gross violation of Condition II, but when the capacity is drawn as in Fig. 4, it is seen that this is not the case. If Fig. 3 had a more complex structure, its equivalence to Fig. 4 may not be at all clear.

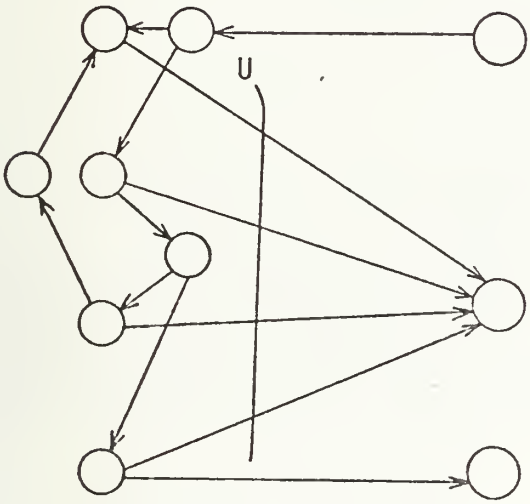


Fig. 3

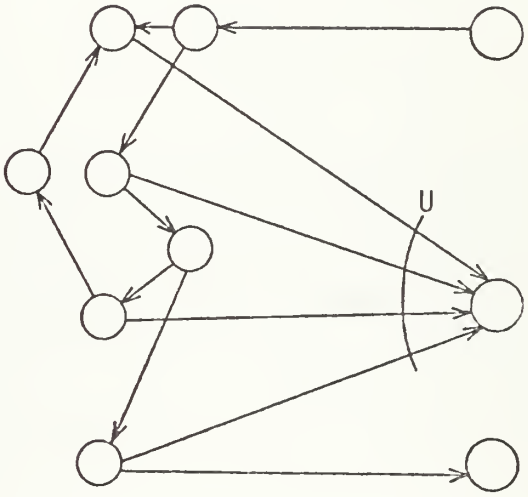


Fig. 4

This approach can be particularly helpful in problems with subnetworks which have transportation structures, where the known sum of flows at any node can considerably simplify a joint capacity. For example, consider the equivalent capacities in the identical networks in Figs. 5 and 6. Note that Fig. 5 appears to violate Condition II. Fig. 6 is obtained by noting that in Fig. 5, U includes <u>all</u> arcs incident with the pure demand nodes 3 and 4. Since the flows along these arcs must total <u>exactly</u> $b_3 + b_4$, then the remaining arc (1,5) covered by U must then have a capacity $U-(b_3 + b_4)$.

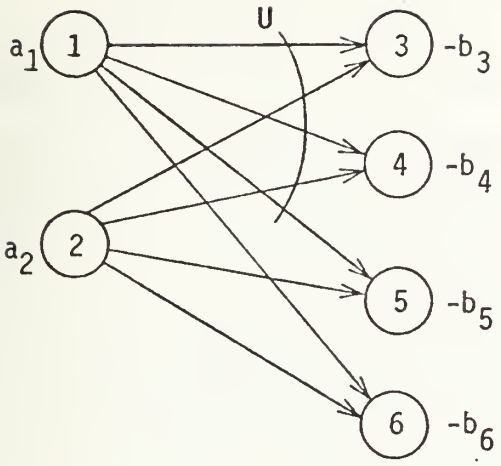


Fig. 5

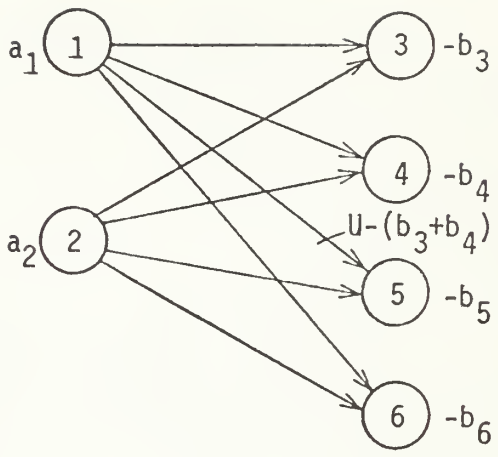


Fig. 6

In the rest of the thesis, a joint capacity in any problem is assumed to satisfy conditions which ensure a feasible solution, and also conditions which ensure that the capacity is non-redundant (i.e., binding).

D. THE TRANSFORMATIONS

1. Transformation I

This Transformation reverses the direction of a capacitated arc.

Suppose (1,2) is an arc in a transshipment problem, with unit cost of flow c, and such that $x_{12} \leq U$. Each of nodes 1 and 2 may have demand or supply T_1, T_2. See Fig. 7.

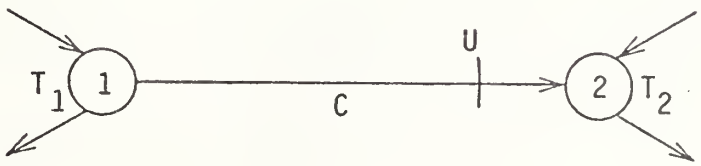


Fig. 7

Consider the equivalent network in Fig. 8, which can be regarded as the result of introducing a counter-flow of magnitude U in the arc, which gives a resultant flow of $x_{21} = U - x_{12}$ in the direction opposite to the original flow x_{12}, and making the cost negative.

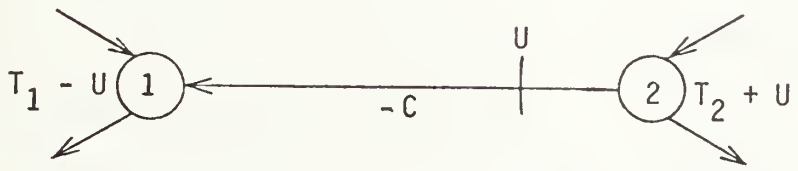


Fig. 8

Algebraically, this is simply equivalent to substituting $U - x_{21}$ in place of x_{12} wherever it occurs in the flow conservation equations and objective function of the original problem.

	Original Problem	Transformed Problem
Node 1 flow conservation:	$\ldots -x_{12} + T_1 = 0$	$\ldots -(U - x_{21}) + T_1 = 0$
		or $\ldots + x_{21} + (T_1 - U) = 0$
Node 2 flow conservation:	$\ldots +x_{12} + T_2 = 0$	$\ldots +(U - x_{21}) + T_2 = 0$
		or $\ldots -x_{21} + (T_2 + U) = 0$
Arc capacity:	$0 \leq x_{12} \leq U$	$0 \leq U - x_{21} \leq U$
		or $U \geq x_{21} \geq 0$
Objective function:	$\ldots cx_{12} + \ldots$	$\ldots +c(U - x_{21}) + \ldots$

The solution to the original problem is obtained from the solution to the transformed problem as follows:

a. $x_{12} = U - x_{21}$. Other flows are identical between the two problems.

b. Add cU to the objective of the transformed problem.

Transformation I is of little interest by itself, but is useful as part of a more complex transformation.

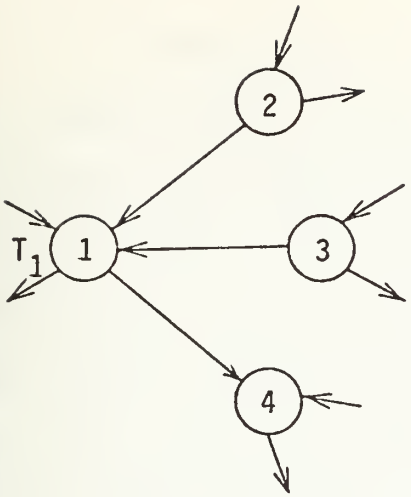
2. Transformation IIa and IIb

Given a joint constraint over a subset of the arcs incident to a node, this Transformation (actually a sub-class of transformations) transfers the capacity to a single (artificial) zero-cost arc, leaving all other arcs uncapacitated.

To avoid violating Condition III, the joint constraint must be imposed on the algebraic sum of flows in the arcs. Flows must therefore have attached signs appropriate to their orientation with respect to the incident node.

Constraints having sums which include flows with opposite signs must be clearly specified as to whether they are permitted to have negative lower bounds.

Consider the transshipment network in Fig. 9, with a joint constraint as shown. Note that a constraint which includes the sum $x_{21} + x_{31} + x_{41}$, for example, is not permitted for the reason given above. The equivalent network is shown in Fig. 10.



with constraint:

$0 \leq x_{21} + x_{31} - x_{14} \leq U$

Fig. 9

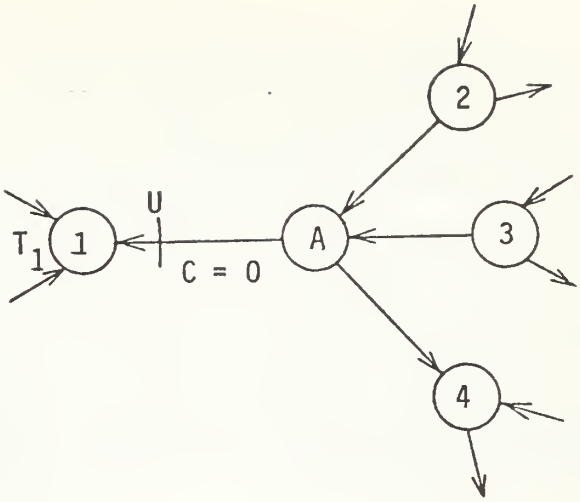


Fig. 10

A positive lower bound is also permitted, with or without an upper bound.

If $x_{21} + x_{31} - x_{14}$ is not constrained to be non-negative, Fig. 11 results. The artificial arc pair is necessary to allow the sum $x_{21} + x_{31} - x_{14}$ freedom to flow in either direction (i.e., according to its sign) between nodes 1 and A.

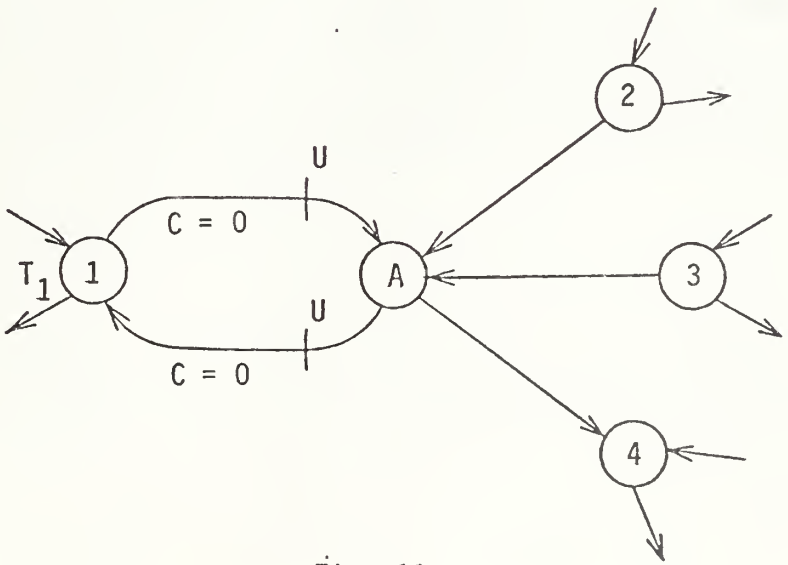


Fig. 11

A constraint of the form typified by $x_{21} + x_{31} - x_{14} \leq 0$, when written as $x_{21} + x_{31} \leq x_{14}$, can be easily seen to be useful, for example in a road network where the total incoming traffic along some subset of roads at a busy town is required to be no greater than total outgoing traffic along some other subset of roads, while the more general form $x_{21} + x_{31} - x_{14} \geq U$ would require a positive difference between the two sets of traffic flows.

Transformation IIb is a special case of IIa, and is commonly known as the "capacitated node." It arises when an upper (and/or lower) bound is imposed on the amount of flow passing through a node, which, in the case of a pure transshipment node, is equivalent to putting a joint capacity on all the arcs entering (or leaving) the node. Transformation IIa can then be applied directly. For a source or sink node, the demand or supply at the node may be included as part of the total flow passing through. Figs. 12 and 13 show the Transformation for a source node with an upper bound U, which includes the supply as part of the total. If the supply is not to be included, then the supply a_1 is placed at node 1 instead of node A.

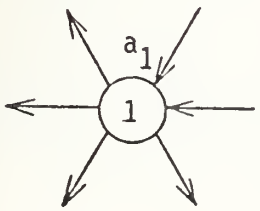


Fig. 12

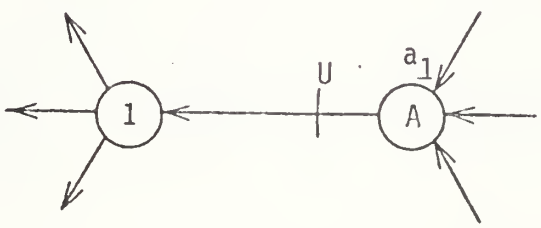


Fig. 13

A set of jointly capacitated arcs at a node is a special case of a jointly constrained set, when all arcs in the constrained set are oriented in the same direction.

3. Transformations IIIa and IIIb

Given a capacitated arc, these Transformations consist of Transformations II and I applied sequentially to create an uncapacitated transshipment structure with one additional artificial arc and node.

Consider the capacitated arc (1,2) in a transshipment network, Fig. 14. Applying Transformation II gives Fig. 15.

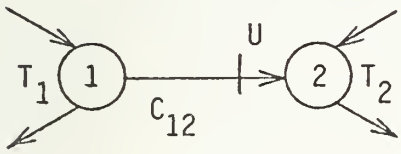


Fig. 14

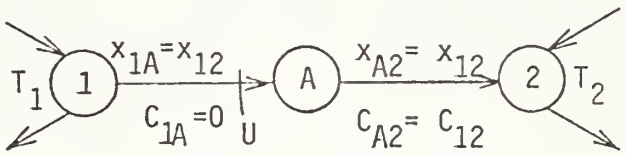


Fig. 15

Applying Transformation I to reverse the arc (1,A) gives Fig. 16. It is seen immediately from flow conservation at node A that flows $x_{A1} = U - x_{A2}$ and $x_{A2} = x_{12}$ cannot exceed U. Thus, the redundant capacity on arc (A,1) can be removed, with the end result in Fig. 17. Call this Transformation IIIa.

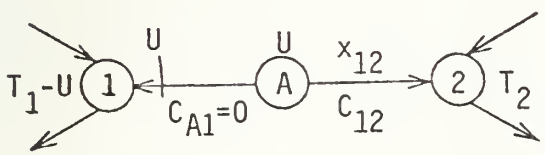


Fig. 16

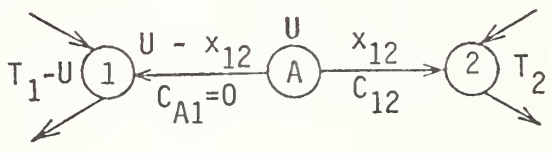


Fig. 17 (Transformation IIIa)

No modification to the objective function is necessary to obtain the solution to the original network. The original flow x_{12} is found directly in arc (A,2).

Alternatively, the original network may be transformed to that shown in Fig. 18, instead of Fig. 15. Reversal of arc (A,2) with Transformation I, and using the same reasoning as above, gives the end result in Fig. 19, which has reversed arcs compared to Fig. 17, and the original flow x_{12} is found directly in arc (1,A).

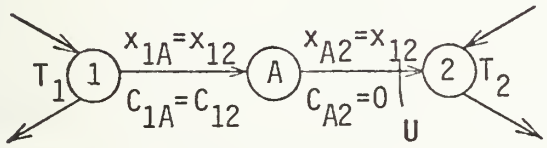


Fig. 18

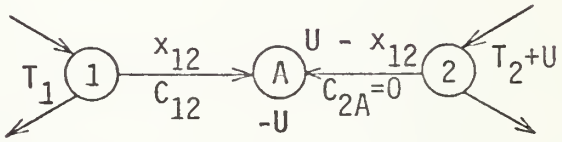


Fig. 19 (Transformation IIIb)

It is noted that the two end results typified in Figs. 17 and 19 may be transportation structures, if the demand/supply and arc orientations at nodes 1 and 2 are appropriate, since node A is already either a pure source or sink node in the transformed networks.

4. Transformation IV

This Transformation, applied to a jointly capacitated subset of arcs at a node in a capacitated transportation network, produces an uncapacitated transportation network with the addition of two artificial nodes and arcs.

Suppose we have part of a transportation network as in Fig. 20. Application of Transformation II yields the equivalent transshipment network in Fig. 21.

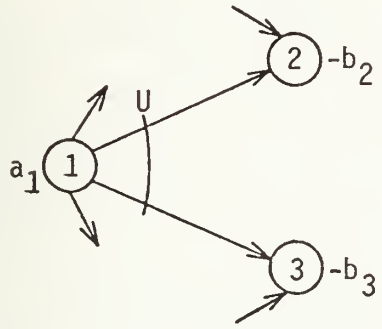


Fig. 20

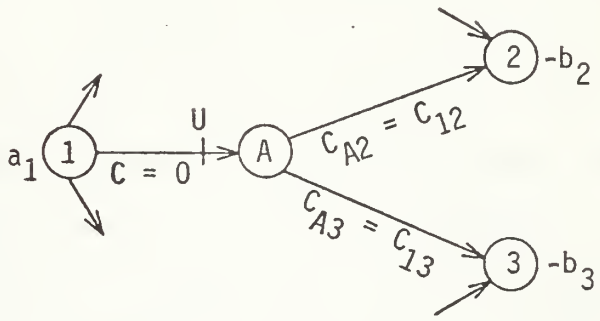


Fig. 21

Transformation IIIb applied to arc (1,A) gives the network in Fig. 22, which can be seen to be part of a transportation network when re-arranged as in Fig. 23.

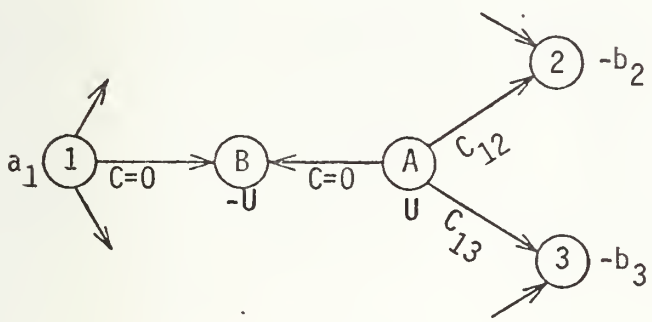


Fig. 22

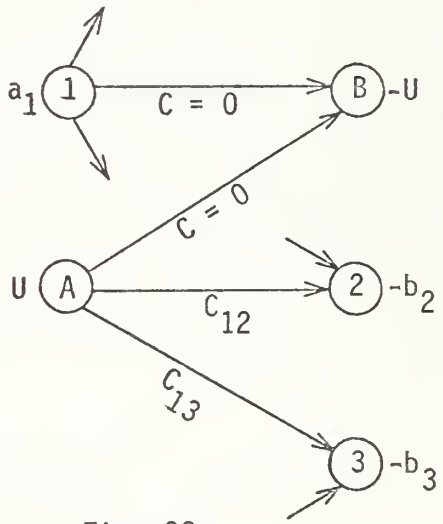


Fig. 23

No modification to the objective function is necessary, and the original flows x_{12}, x_{13} are found in arcs (A,2), (A,3) respectively.

Suppose the original problem had a jointly capacitated set of arcs entering a sink. Transformation II, and then

Transformation IIIa (instead of IIIb), would produce a "mirror image" to Fig. 23 above.

This Transformation is of course applicable to individually capacitated arcs. If all the arcs incident at a node i in a transshipment network have a joint capacity U, then for U to be binding, it is required that $U < a_i$. This case makes sense only if it occurs in a network with total supply $\Sigma a_i >$ total demand Σb_j, otherwise $U < a_i$ makes the problem infeasible. Additionally, only a source node may be capacitated in this way, since a binding constraint $U < b_j$ at a sink immediately causes infeasibility (the case where $\Sigma b_j > \Sigma a_i$ is not considered). In this special case (Fig. 24), a_1 may be replaced directly by U to give the uncapacitated Fig. 25, which requires no artificial arcs or nodes. However, in the standard procedure for the case of $\Sigma a_i > \Sigma b_j$, where all the sources are connected at zero cost to a dummy sink with demand $-(\Sigma a_i - \Sigma b_j)$, this Transformation for the special case in Fig. 24 necessitates substitution of U for a_1 in the demand $-(\Sigma a_i - \Sigma b_j)$.

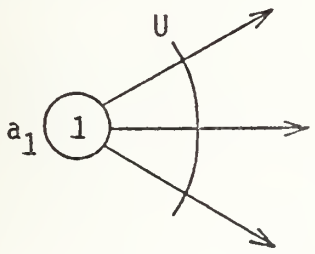


Fig. 24

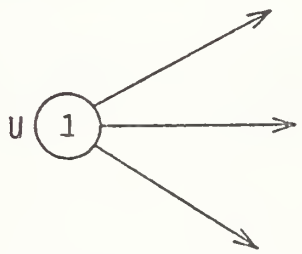


Fig. 25

5. Transformation V

A network with a node at which one joint capacity is "nested" within another in the manner described in Condition I

in Section II.C, is transformed to a transshipment network which includes two individually capacitated arcs. In this form it may be used directly with suitable codes. If the original network was a transportation structure, the transformation may be carried further to give an uncapacitated transportation network.

Suppose we have part of a transshipment network as in Fig. 26. Necessarily, $U_1 > U_2$ for feasibility and non-redundancy of U_2. Application of Transformation II to arcs capacitated by U_1, and then again to arcs capacitated by U_2, gives Fig. 27.

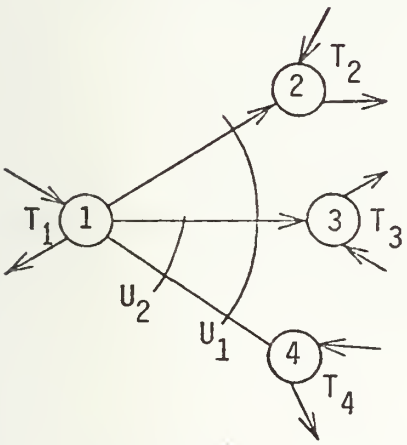


Fig. 26

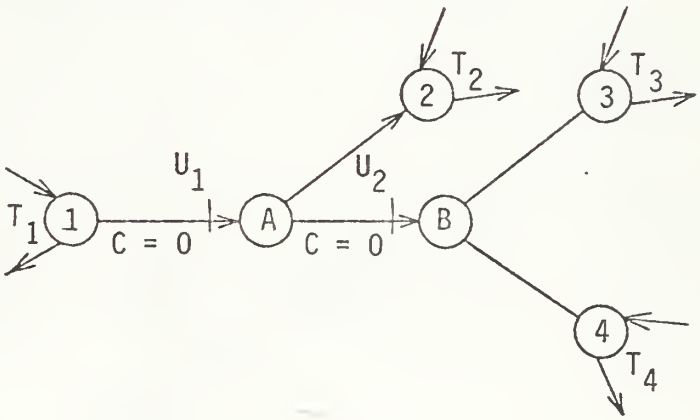


Fig. 27

Suppose now that the original problem was part of a transportation network, and it is desired to obtain an uncapacitated transportation network. Beginning with Fig. 28 which is analogous to Fig. 27, application of Transformation IIIb to arcs (1,A) and (A,B) gives Fig. 29, which can be arranged in the usual bipartite form as in Fig. 30.

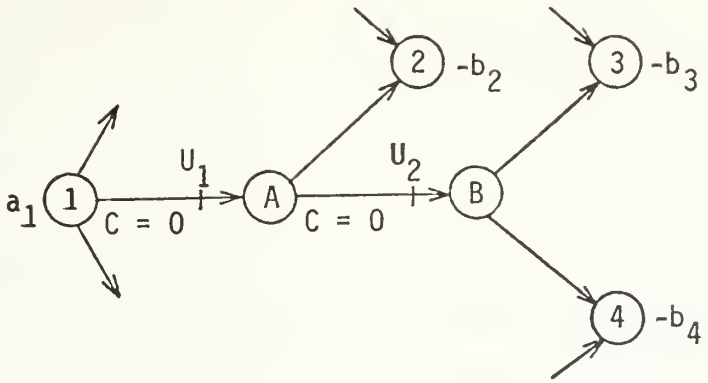


Fig. 28

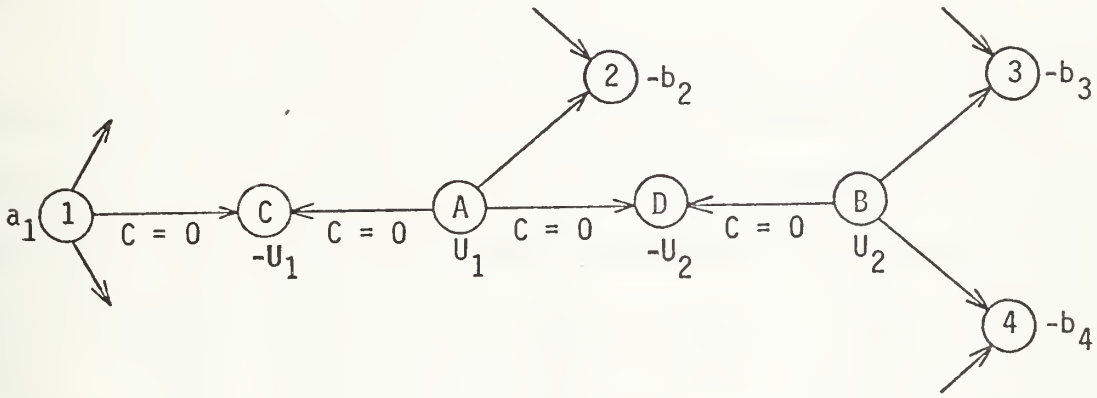


Fig. 29

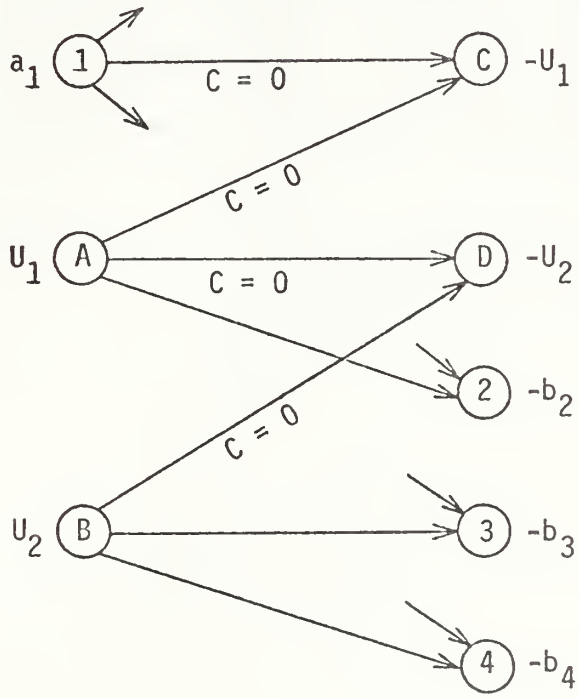


Fig. 30

In either Fig. 27 or Fig. 30, no change to the objective function is necessary, and any original flows x_{ij} are found in that arc which enters node j in the transformed problem.

6. Transformation VI

This Transformation is one means of formulating a transshipment problem as a transportation problem. It is found, for example, in Wagner [9].

In this transformation, every node in the transshipment problem which has all its incident arcs either entering or leaving (i.e., is either a pure sink or source node) is left unchanged. The following transformation is then applied to every transshipment node, a typical example of which is in Fig. 31. The node A can be split into two nodes, one with all the original arcs leaving (node A1), and the other with all original arcs entering (node A2). A zero-cost arc (A2, A1) is added as in Fig. 32, which is clearly equivalent to Fig. 31.

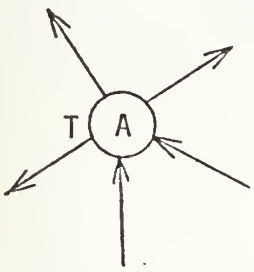


Fig. 31

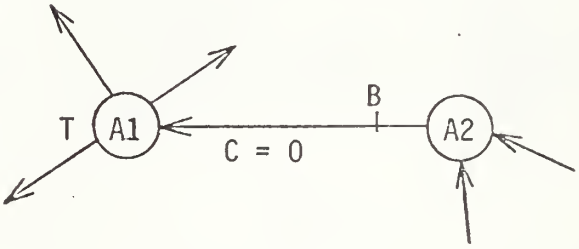


Fig. 32

In anticipation of using Transformation I, a superfluous capacity is imposed on arc (A2, A1). A safe capacity to use is obviously the sum of all supplies in the problem (or any larger number), since no sum of flows in the set of

arcs entering (or leaving) any node can exceed this number, which is denoted by B, and given the interpretation of a "buffer stock at node A" in Wagner [9].

Applying Transformation I to reverse arc (A2, A1), and hence obtain a bipartite structure, we obtain Fig. 33. At node A2, the flow conservation equation is now $x_{A1,A2} = B - \sum_{i \neq A1} x_{i,A2}$, which ensures $x_{A1,A2} \leq B$. The constraint on arc (A1, A2) is therefore superfluous, and is removed to give finally Fig. 34, in which A1 and A2 are pure source and sink nodes respectively. Application of this transformation to every transshipment node, together with the unchanged pure source and sink nodes in the original problem, will clearly yield an equivalent transportation problem in which the original flows are directly identifiable and the objective function unaffected by the introduction of zero-cost artificial arcs such as (A1, A2).

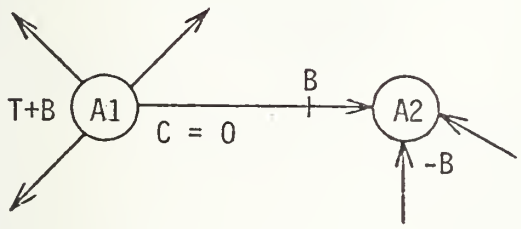


Fig. 33

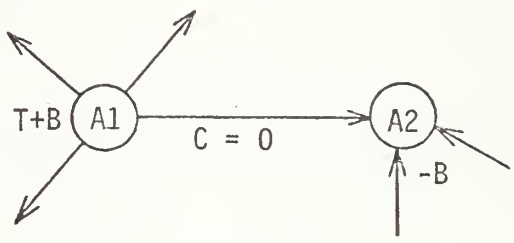


Fig. 34

The above transformation is not affected by capacities, joint or individual, on the arcs in the original problem. Capacities in the original problem are imposed on the corresponding arcs in the transformed problem.

7. Transformation VII

This Transformation is applicable to a pure source node which delivers flow to precisely two other nodes. It eliminates the source node and its incident arcs, at the cost of introducing an artificial capacitated arc. It can therefore be used to reduce the number of nodes and arcs in problems with the special structure mentioned above, if a code capable of handling capacitated transshipment problems is available.

Suppose the network in Fig. 35 is part of a transshipment problem. This is re-drawn in Fig. 36, with a superfluous capacity a_1 imposed on arc (1,2).

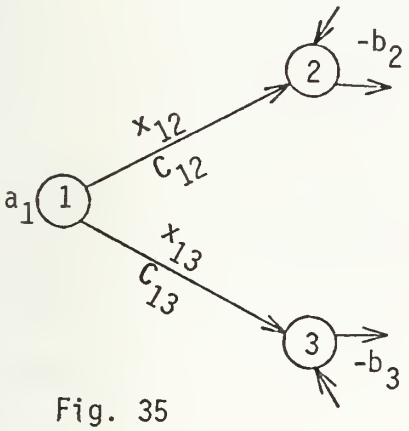


Fig. 35

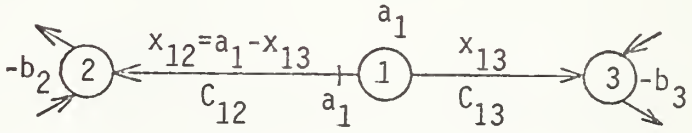


Fig. 36

Applying Transformation I to reverse arc (1,2) yields Fig. 37, which is algebraically simplified in Fig. 38.

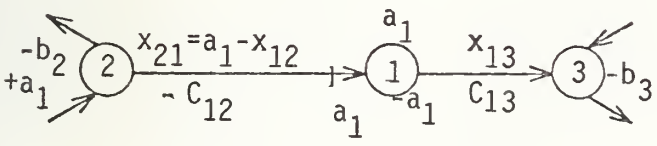


Fig. 37

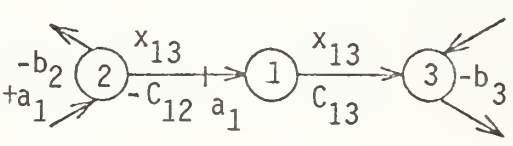


Fig. 38

Node 1 in Fig. 38 is clearly superfluous, and its removal yields finally Fig. 39 which is equivalent to the original problem provided:

a. x_{13} in the original problem is obtained from x_{23} in the transformed problem.

b. x_{12} in the original problem is obtained by subtracting x_{23} from a_1.

c. The original objective function contained costs $c_{13}x_{13} + c_{12}x_{12}$. The transformed objective function contains $x_{13}(c_{13} - c_{12}) = c_{13}x_{13} - c_{12}x_{13} = c_{13}x_{13} - c_{12}(a_1 - x_{12}) = c_{13}x_{13} + c_{12}x_{12} - a_1c_{12}$. Thus the original cost is obtained by adding a_1c_{12} to the objective function of the transformed problem.

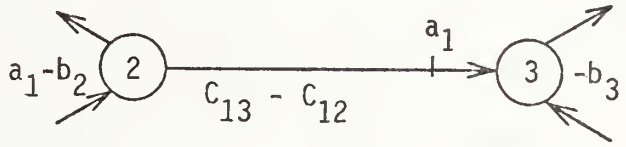


Fig. 39

This transformation can be interpreted as if all of supply a_1 were delivered first to node 2, at cost a_1c_{12}. Subsequently, each unit increase in x_{23} (= x_{13} in the original problem) will increase cost by c_{13} units and decrease cost by c_{12} units. This can be seen in Fig. 3 to be precisely the same mechanism for a unit increase in x_{13} in the original problem, and arises because flow conservation at node 1 forces a unit decrease in x_{12} for every unit increase in x_{13}.

8. Transformation VIII

This Transformation is applied to effectively remove (i.e., reduce to zero) a lower bound on the flow along an arc.

Suppose we have arc (1,2) in Fig. 40 with both an upper and lower bound. Since at least L units must be delivered

along the arc, we can simply apply a tranformation which assumes that L units have already been delivered, leaving the balance $x_{12} - L$ (≥ 0) to be determined by the solution algorithm. The upper bound has also to be correspondingly modified as shown in Fig. 41, where x'_{12} refers to flow in the transformed network.

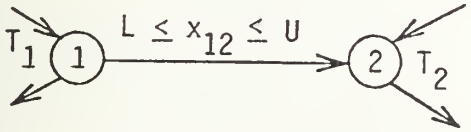


Fig. 40

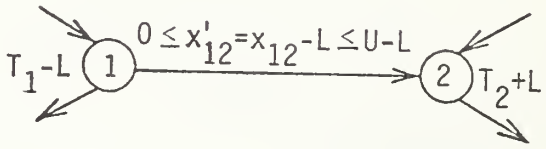


Fig. 41

The original flow x_{12} is obtained from $x'_{12} + L$, and the original objective function value by adding Lc_{12} to the optimal solution of the transformed network.

In the extreme case when U = L, equivalent to specifying $x_{12} = U$, we can eliminate arc (1,2) to obtain Fig. 42. Suitable operations must of course be carried out to obtain the original solution.

Fig. 42 When U=L

9. Transformation IX

This transformation is applicable to a pure source node which delivers flow to two nodes via capacitated arcs. It reduces the number of capacitated arcs to one, and thus may be useful in reducing the complexity of a given problem.

Suppose Fig. 43 is part of a transshipment network. We have $x_{13} \leq U_{13} <=> a_1 - x_{13} \geq a_1 - U_{13} <=> x_{12} \geq a_1 - U_{13}$. The arc (1,3) can therefore be uncapacitated by imposing a lower bound on flow in arc (1,2), resulting in Fig. 44.

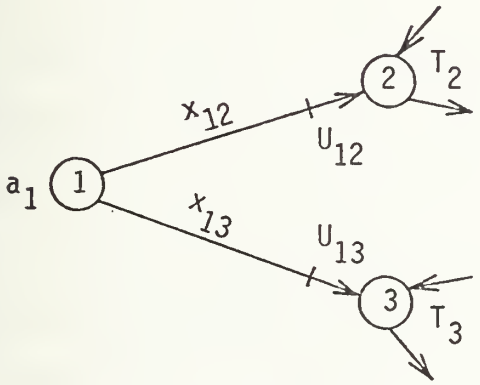


Fig. 43

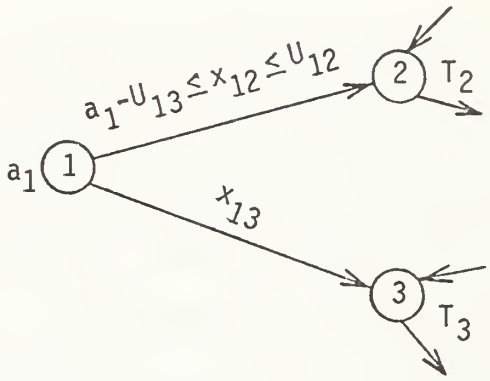


Fig. 44

Application of Transformation IX results in Fig. 45 with only one capacitated arc. The original flow x_{12} is obtained from $x'_{12} + a_1 - U_{13}$, and the original objective function value by adding $(a_1 - U_{13})c_{12}$ post-optimally in the transformed problem.

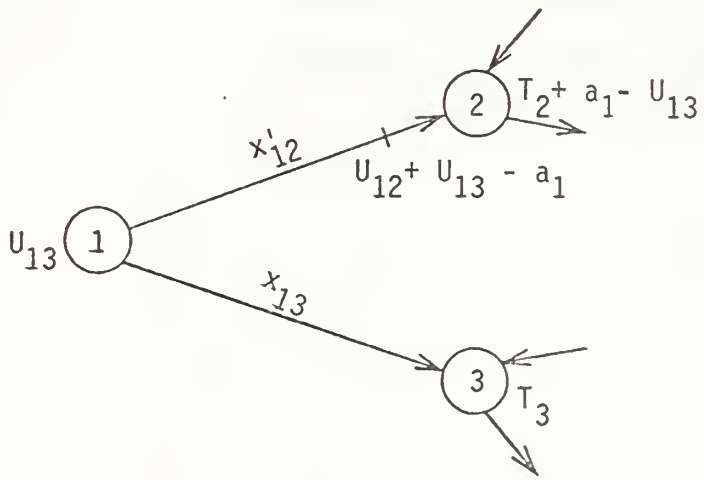


Fig. 45

III. TWO SPECIAL CASES OF THE MULTICOMMODITY FLOW PROBLEM

A. GENERAL DISCUSSION

When viewed as an extension to the capacitated single-commodity transshipment model, the multicommodity model has naturally wider applicability in more complex and realistic situations. Apart from the more obvious applications of shipping different goods through a distribution system, it is sometimes necessary to keep track of groups of items which may be physically identical. For example, some traffic assignment problems require that the identity of travellers going between various node pairs must be kept. The model is used in problems which require finding optimal flow patterns (quantity of flows and routes) for each of a number of distinguishable commodities in a capacitated network structure. The criteria of optimality is minimum total cost, which includes the maximum sum of flows criteria as a special case. A capacity on an arc takes the form of an upper bound on the sum of flows of all commodities along that arc. Only directed arcs are considered in the following results, since a capacitated undirected arc in a multicommodity problem does not in general have an equivalent network formulation because, even if the usual single-commodity formulation of two jointly-capacitated arcs oriented in opposite directions between the two incident nodes is resorted to, there will generally be flow (of different commodities) in both arcs, unlike in the single commodity case.

For any one of the commodities, the flow in only one of the two arcs can be at a non-zero level.

The general minimum cost multicommodity transshipment problem with r commodities is usually formulated as follows:

Minimize $\sum_{k=1}^{r} \sum_{i} \sum_{j} c_{ij}^{k} x_{ij}^{k}$,

subject to:

$$\sum_{j} x_{ij}^{k} - \sum_{\ell} x_{\ell i}^{k} = T_{i}^{k} \quad \text{, for each } i, k,$$

$$\sum_{k=1}^{r} x_{ij}^{k} \leq U_{ij} \quad \text{, for each } i, j,$$

and

$$x_{ij}^{k} \geq 0 \quad \text{, for each } i, j, k.$$

Additionally, it is assumed that $\sum_{i} T_{i}^{k} = 0$ for each k.

The general multicommodity problem above is non-unimodular (thus, even with integer T_{i}^{k}, U_{ij}, the solution will in general be fractional) and is solved as a linear programming problem. For large problems, therefore, the comparative slowness of L.P. techniques is again a limiting factor, more so because an r-commodity problem is approximately r times the size of a single-commodity problem with the same node-arc structure.

In each of the following two subsections is provided (as part of a constructive derivation) a transformation of a special case of the multicommodity problem into an equivalent single-commodity capacitated transshipment problem. Each of these two transformed networks (or with further transformation into equivalent uncapacitated transshipment/transportation networks) may then be solved directly and advantageously with network-based solution codes, yielding integer solutions.

Specifically, the two results are:-

a. A network-based derivation and significant extension of a result by Rebman [7] which originally considered the 2-commodity transportation problem where all capacitated arcs are incident with a single node in the network structure,

b. A network-based simplified derivation of a result by Evans, Jarvis and Duke [4], which considered the r-commodity transportation problem with 2 sinks (or 2 sources).

The original results were obtained by using matrix theory. It is believed that the derivations below will show clearly the advantages of the network approach with respect to simplicity, visual clarity, and greater appeal to intuition. They also illustrate the use of network transformations as part of what amounts to a proof of unimodularity.

B. RESULT I--THE "BUSY NODE" PROBLEM

Rebman [7] showed that an equivalent single-commodity network formulation exists for the 2-commodity transportation problem in which the capacitated arcs from a tree with at most one interior node (i.e., all capacitated arcs in the problem are incident with a common node--the "busy node," which is either a source or sink node).

It will be shown here that his result can be extended in the following ways:-

a. $r \geq 2$ commodities.

b. The network may be a transshipment network, in which the "busy node" may be any node (source, sink, or pure transshipment).

c. Multiple arcs between nodes are permitted.

d. Permissible arcs need not be identical for each commodity. A permissible arc for the k^{th} commodity is defined

as an arc along which the commodity is permitted to flow at a positive level at a finite cost.

e. Joint capacities and lower bounds may be imposed on subsets of the set of arcs incident with the busy node, provided the three conditions in Section II.C. for an equivalent network to exist are satisfied. This includes the requirement that each member of a capacitated subset of arcs be oriented in the same direction with respect to the busy node, for every commodity. However, disjoint capacitated subsets of arcs may be oriented in opposite directions.

f. It is not necessary that a capacity on an arc applies to the sum of flows of <u>all</u> commodities along it--i.e., a capacity on the sum of flows of some subset of the r commodities is permitted, the rest of the commodities either being delivered without bound along the arc, or having other joint capacities imposed on nested or disjoint subsets of commodities.

There is of course no requirement that <u>all</u> arcs incident with the busy node be capacitated. It is required that equality of total supply to total demand holds for each commodity.

The result has obvious application to multicommodity transshipment problems with one particular central terminal which requires incident routes to be capacitated in order to minimize congestion. Pictorially, a typical busy node could appear as in Fig. 46.

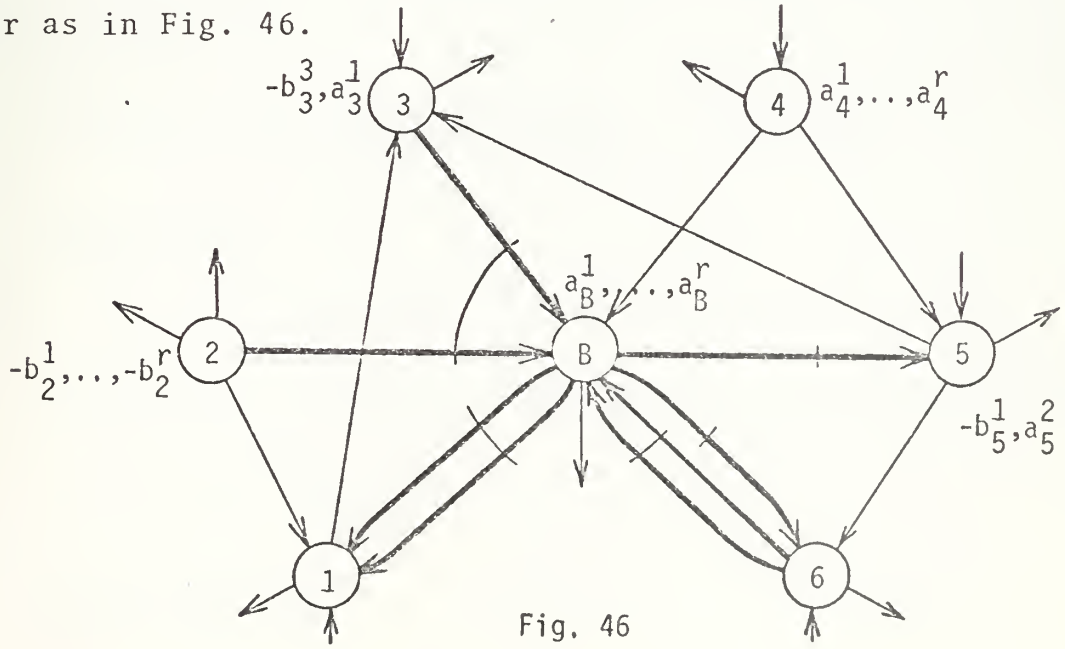


Fig. 46

The busy node B shown above is a source transshipment node. The heavy-lined arcs are capacitated, with lines drawn across groups of arcs which have joint capacities.

Algebraically, the problem is formulated below, using the following notation:

N	The set of nodes of the network.
A	The set of arcs of the network, defined as node pairs.
A^k	$\subseteq A$, the set of permissible arcs for the k^{th} commodity.
$A(i)$	$\subseteq A$, the set of arcs incident to node i.
$A(i,j)$	$\subseteq A$, the set of multiple arcs from node i to node j.
$A[B]$	$\subseteq A(B)$, the set of capacitated arcs incident to busy node B.
arc $(i,j;\ell)$	$\in A(i,j)$, the ℓ^{th} multiple arc from node i to node j.
$C^k_{ij\ell}, x^k_{ij\ell}$	Unit cost and flow of k^{th} commodity along arc $(i,j;\ell) \in (A^k \cap A(i,j))$. If the third subscript is absent, it is understood that there is only one arc (i,j).

The problem is:

$$\text{Minimize } \sum_{k=1}^{r} \Big[\sum_{(i,j;\ell) \in A^k \cap A(i,j) \text{ such that}} C^k_{ij\ell} x^k_{ij\ell} \Big],$$

subject to:

$$x^k_{ij\ell} \geq 0, \quad \text{all } i, j, \ell, k,$$

$$\sum_{j,\ell \,|\, (i,j;\ell) \in A^k \cap A(i)} x^k_{ij\ell} - \sum_{n,m \,|\, (n,i;m) \in A^k \cap A(i)} x^k_{nim} = T^k_i, \quad \text{all } i, k,$$

and at node B,

$$\sum_{k \mid (i,j;\ell) \in A^k \cap A[B]} x^k_{ij\ell} \leq U_{ij\ell} \ , \quad \text{each } (i,j;\ell) \in A[B]$$

To avoid further notational complexity, the possibility of joint capacities or capacities which apply to only some commodities along an arc are not reflected in the above formulation. These extensions are dealt with later.

1. Constructive Derivation of the Result

Suppose r = 3 for a transshipment network, part of which is shown in Fig. 47. Busy node B is in this example a sink transshipment node with demands b^k_B. For clarity, only those nodes directly joined to B are shown, and in fact only these nodes and the arcs joining them to B are visually relevant to the proof. Individually capacitated arcs are used for simplicity. We consider here the case where permissible arcs are identical for each commodity, and the capacities apply to the sum of flows of all commodities along an arc. It is seen later that the proof is unaffected by these simplifying assumptions.

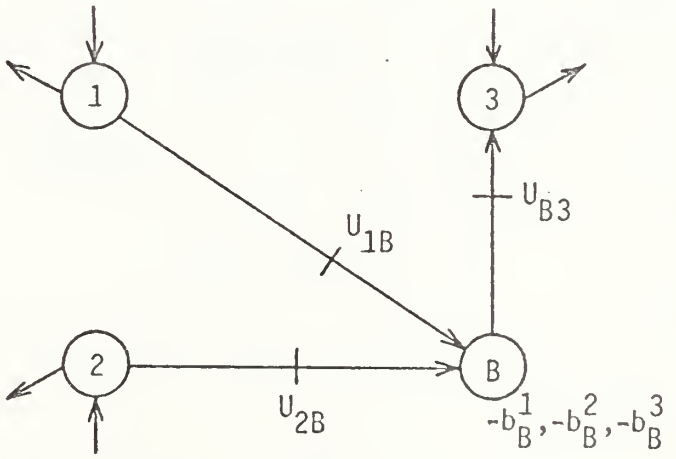
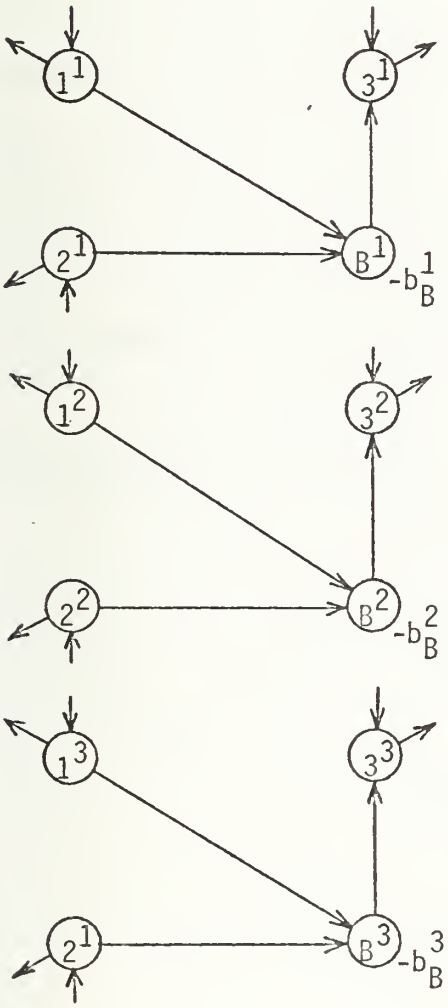


Fig. 47

The network is equivalent to r = 3 separate networks, one for each commodity, and each identical in structure to the original network, with capacities applied to appropriate arcs across the r networks. Let node i^k, in the network corresponding to the k^{th} commodity, correspond to node i of the original problem. Fig. 48 results. We now have essentially identical subnetworks coupled by capacity constraints.



with capacities:

$$\sum_{k=1}^{3} x_{1B}^k \leq U_{1B} ,$$

$$\sum_{k=1}^{3} x_{2B}^k \leq U_{2B} ,$$

$$\sum_{k=1}^{3} x_{B3}^k \leq U_{B3} .$$

Fig. 48

In the standard L.P. formulation, the constraint matrix of the whole problem is in block diagonal form, with the blocks corresponding to the constraint matrices of the subnetworks

in Fig. 48 and coupled by rows expressing the capacity constraints.

It is well known for the single-commodity transshipment problem with equality of total demand to total supply that any one of the node flow conservation equations (corresponding to rows in the L.P. constraint matrix) is linearly dependent on the other conservation equations. This is equivalent to saying that by conservation of flow over the whole network, given supplies and demands at all the other nodes, the net flow at a node i must equal T_i with due regard to signs. For a sink node i, this means that given demand and supply in the rest of the network, the net amount of commodity deposited at the node is predetermined to be b_i.

The above line of reasoning is true for each of the r subnetworks in Fig. 48.

Suppose, therefore, in each of those subnetworks, we choose node B^k to be the node with the linearly dependent equation. If we re-combine all B^k back into a single node B with demand $\sum_{k=1}^{r}(-b_B^k)$, flow conservation within each subnetwork will compel an amount b_B^k to be deposited at B, which is precisely what the original problem requires. The result is at Fig. 49, which is a single-commodity transshipment network with joint capacities over appropriate arcs incident to B. The capacities satisfy the conditions in paragraph 9 of the previous Section.

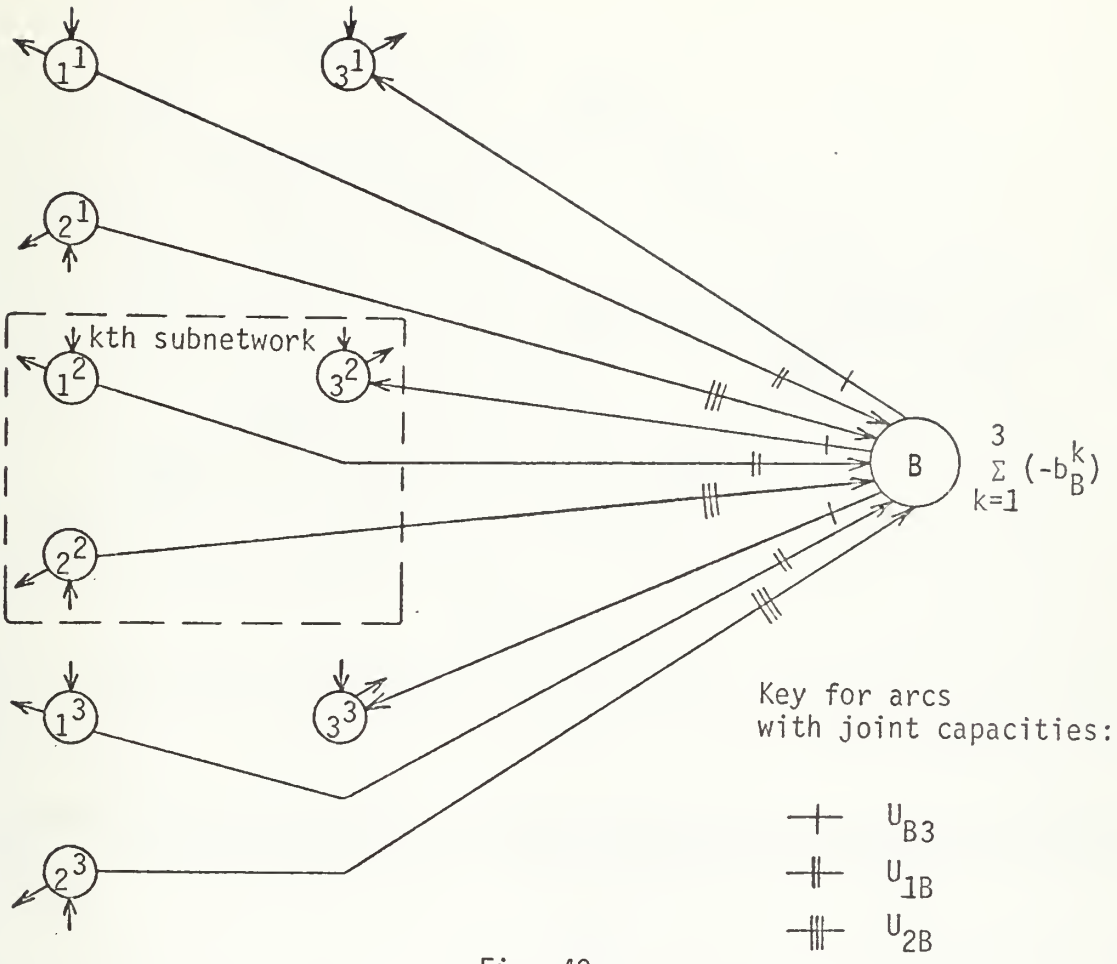


Fig. 49

Fig. 49 may then be reduced to the standard capacitated transshipment network in Fig. 50, using Transformation II. The original flows are directly identifiable and no change to the objective function is needed. If necessary, other Transformations may be applied to uncapacitate it or convert it to a transportation network.

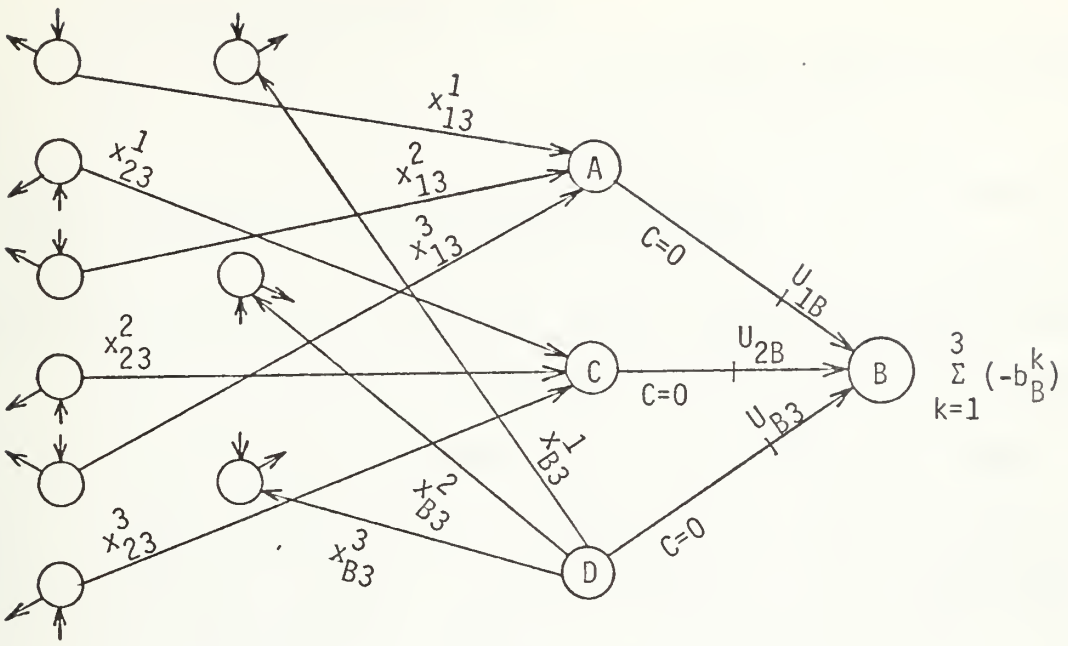


Fig. 50

Note that the above construction is valid, independent of:

a. Whether the subnetworks (and hence the original problem) have a transshipment or transportation structure.

b. The presence of multiple arcs between any node and node B.

c. Whether the subnetworks are identical or not (except of course for nodes B^k). This allows permissible arcs to be different for each commodity (provided, of course, the permissible arcs for different commodities which are included in a joint capacity between a node pair are oriented in the same direction).

d. The type of nodes (source, sink, or transshipment) involved.

e. The number of commodities.

f. The existence of lower bounds on arcs. If present, these could be removed at the transformation stage depicted in Fig. 49, by using Transformation VIII.

g. The presence of uncapacitated arcs incident with B. These would be dealt with by uncapacitated arcs directly between node B and appropriate nodes in each subnetwork in Fig. 49.

2. Complex Joint Capacities

The description "joint" in the multicommodity context can be taken to apply in two dimensions, because:-

a. For a given arc, the capacity need not include the sum of all commodities flowing along that arc.

b. For a given commodity, the capacity may be imposed over a set of arcs (i.e., jointly, in the single-commodity sense).

If we imagine the subnetworks in Fig. 49 to be stacked up in a pile, then the concept of a two-dimensional joint capacity leads to an interesting visual aid which can be used in small problems to check whether complex joint capacities at the busy node violate conditions for unimodularity.

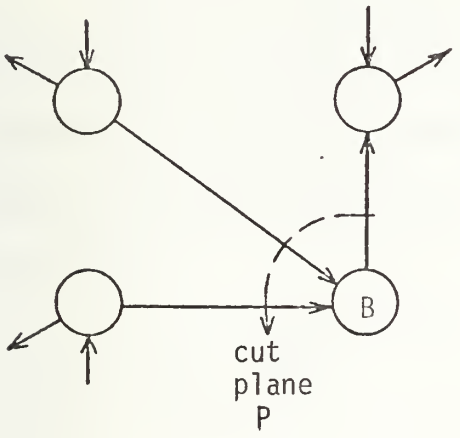


Fig. 51

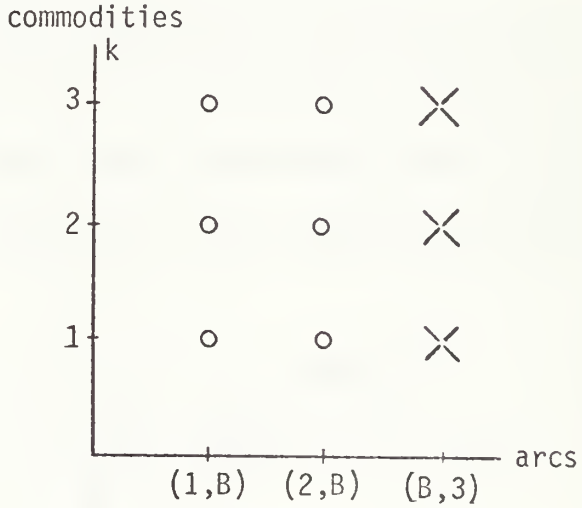


Fig. 52

Suppose Fig. 51 represents the plan view of the stack of subnetworks, and imagine all the arcs corresponding to those incident with node B to be cut at plane P as shown. A cross-section in the plane would appear as in Fig. 52, where the circles represent cut arcs converging, say, into the page to be incident with node B, and the crosses represent arcs

oriented in the opposite direction. Non-permissible arcs for any commodity would not be represented.

Any complex joint capacity over arcs and commodities may be represented simply by sets of appropriate markings in Fig. 52. Such a diagram may be called a "constraint diagram" for the busy node, and may be examined for violation of the conditions set out in Section II.C. Existence of an equivalent network model is not possible, if at any time:

a. A set contains markings indicating arcs oriented in opposite directions, or

b. Two sets overlap (i.e., have an intersection which is a proper subset of both sets).

Fig. 53 illustrates both violations described above, while Fig. 54 illustrates constraints which make a network formulation possible. The geometrical outlines in the figures enclose arc sets which are included in the respective joint capacities.

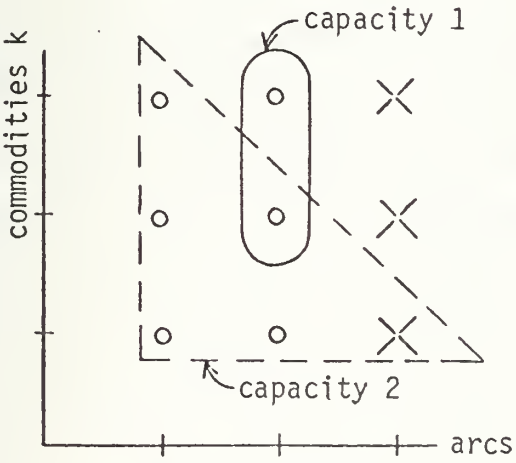


Fig. 53

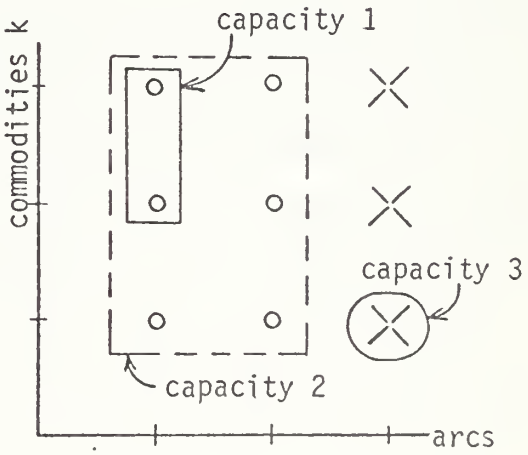


Fig. 54

It should be noted again that arcs represented in these diagrams are not node-disjoint, since they are all

incident with B. Capacities therefore will not violate the Condition III of Section II.C.

For large problems, of course, a systematic computer-executed examination of capacities is more appropriate than such diagrams.

Joint capacities which satisfy conditions for unimodularity in the constraint diagram will also satisfy them when the problem is depicted as a single-commodity problem, as in Fig. 49. Transformations may then be used to simplify complex joint capacities as required.

3. Extensions of the Result

If a multicommodity network has more than one busy node (i.e., has some capacitated arcs which are node-disjoint), it is of interest to determine conditions which allow transformations to an equivalent unimodular single-commodity network.

It is useful to review briefly why the addition of a second busy node would in general make an equivalent single-commodity transformation impossible (if this were possible, the general multicommodity problem would be unimodular). Consider Fig. 49 in the constructive proof of Result I. The basic fact is that the single-commodity formulation makes no distinction between the demands b_B^k. It recognizes only a total demand $\sum_k(-b_B^k)$ at node B of a single hypothesized commodity. What makes the busy node problem work is that flow conservation within the k^{th} subnetwork forces the precise amount b_B^k to be deposited at B. This feature is in general destroyed by the addition of another busy node, say, A. The k^{th}

subnetwork will then by flow conservation deposit a total of $(b_A^k + b_B^k)$ at nodes A and B, but not necessarily b_A^k at node A and b_B^k at node B. An attempt to ensure that precisely b_B^k will be deposited at node B by imposing a joint lower bound of b_B^k on the algebraic sum of flows along the set of arcs between node B and the k^{th} network, for every k, will violate Condition I set out in Section II.C. This will be seen immediately from a constraint diagram for node B.

In looking for extensions, therefore, it is necessary to look for conditions where:

a. The capacities are such that they may be dealt with entirely within a commodity subnetwork.

b. Busy nodes can be "collapsed" into an equivalent single busy node by eliminating arcs, usually at the cost of increasing the complexity of joint capacities and alterations in unit costs along remaining arcs. An appropriate tool for this purpose would be the Transformations (or their inverses) in the previous Section. A simple example is given below, while Result II derived later is a more complex case of "collapse."

c. Delivery of the precise amount of each good can be ensured at each busy node, again usually at the cost of more complex capacitating arrangements.

In attempting (b) or (c) above, it is again necessary to check that capacities which may be added in the attempted transformation do not violate the three Conditions, either between themselves or with existing capacities in the original problem.

An example of (a) above is in the following two closely-related special cases, where transformations to unimodular equivalents exist for capacitated arcs at nodes other than (in addition to) the busy node. These cases arise when:

a. Each capacity at a node is specified to apply only to a particular commodity. The constraint diagram for such a node could appear as in Fig. 55.

b. Joint capacities are applied across all commodities, but permissible arcs for each capacity exist only for one commodity, as in Fig. 56.

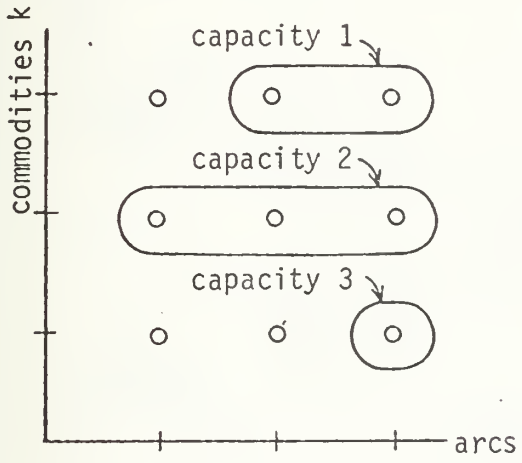


Fig. 55

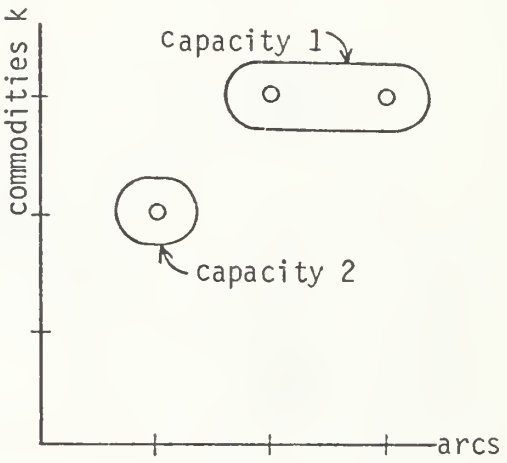


Fig. 56

Each capacity, such as the above at node(s) other than the busy node, are simply dealt with using Transformations, entirely within the commodity subnetwork to which it applies.

A simple example of "node collapse," is as follows. Consider the structure in Fig. 57 below with two busy nodes A and B, and r = 2 commodities, say.

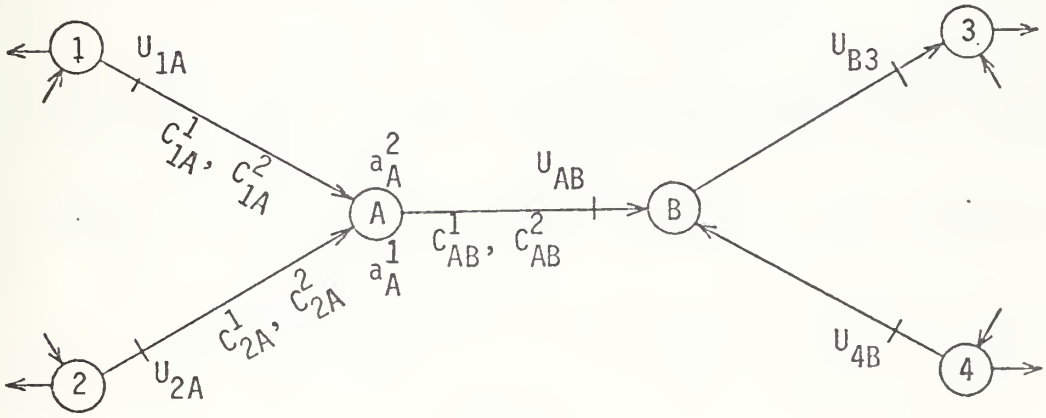


Fig. 57

The structure may be transformed into the equivalent multicommodity structure in Fig. 58, with capacity U_{AB} added

to arcs (1,A) and (2,A) jointly, whose commodity unit costs have C^k_{AB} added on as shown. $a^1_A C^1_{AB} + a^2_A C^2_{AB}$ have to be added to the objective function to bring in the cost of sending supplies at A to B, and flows x^k_{AB} are obtained from $x^k_{1A} + x^k_{2A} + a^k_A$.

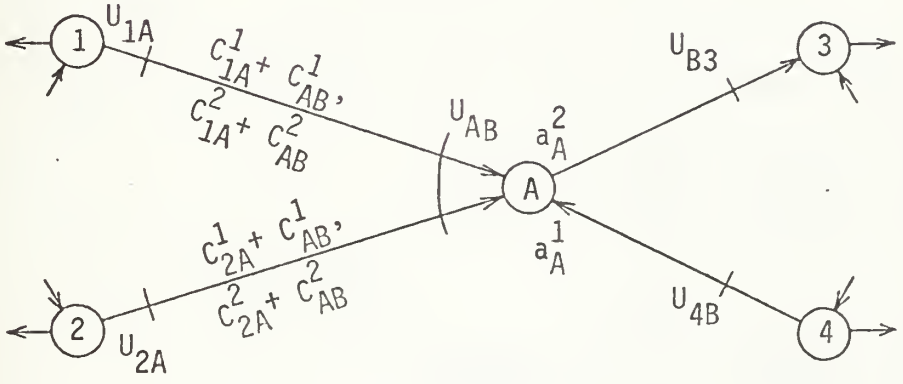


Fig. 58

Fig. 58 is now a single busy node problem with a nested capacitating arrangement which can be transformed using Transformation V, after the problem has been broken up into the usual single-commodity subnetworks.

In transformations of the above kind, it is necessary to see that conditions exist which ensure that the flows x^k_{AB} in the eliminated arc AB are non-negative in the correct direction. For the above example, if either:

a. Arc (1,A) and/or arc (2,A) are reversed, it would yield, for example, $x^k_{AB} = -x^k_{A1} - x^k_{A2} + a^k_A$ (if both are reversed), or

b. Node A became a source node, this would yield $x^k_{AB} = x^k_{1A} - x^k_{A2} - b^k_A$.

In neither case would non-negative of x^k_{AB} be assured after arc AB is removed by node collapse. Note, however, that reversal of arc (B,4) would ensure non-negative x^k_{AB} regardless of arc orientation at A, since in that case $x^k_{AB} = x^b_{B3} + x^k_{B4}$.

If there are demands and/or supplies at both nodes A and B in the original problem, then their algebraic sum is at node A in the transformed problem.

An illustration of conditions which ensure delivery of the required amount of commodity at more than one busy node is as follows. Fig. 59 shows two busy nodes, A and B, part of a multicommodity problem with r = 2, say. Other parts

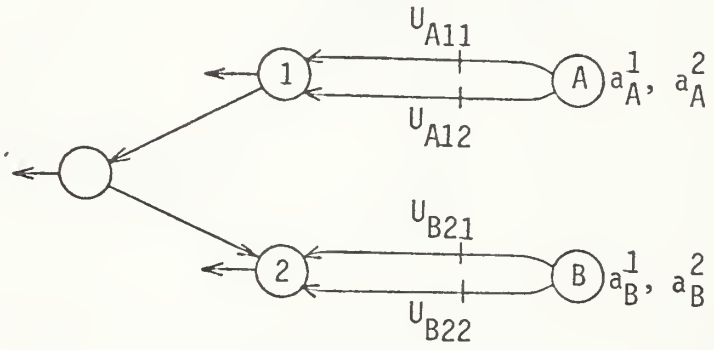


Fig. 59

Fig. 60 shows the problem broken up into commodity subnetworks, with A and B external to the subnetworks, and Transformation II applied to the capacitated arcs.

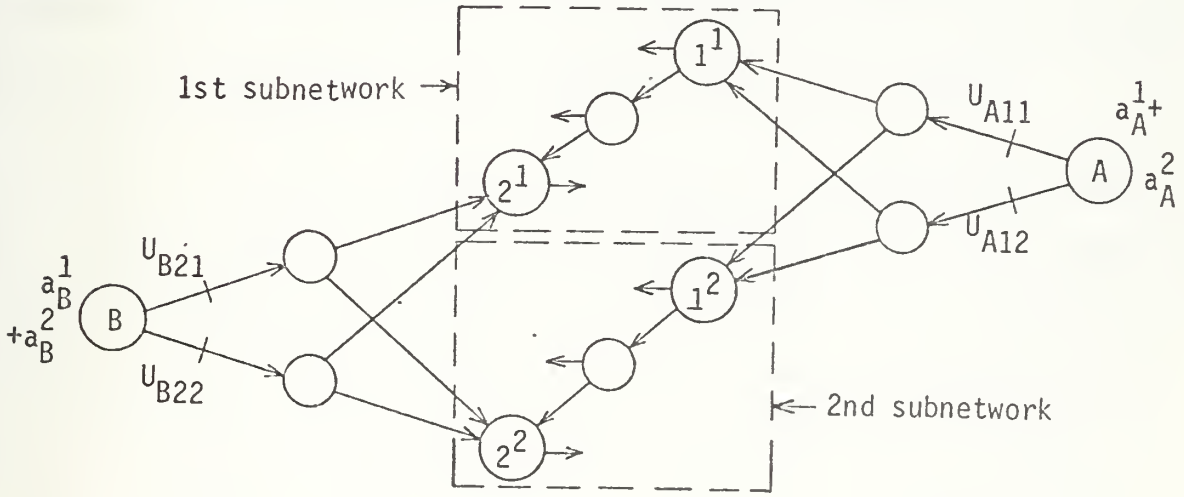


Fig. 60

In Fig. 60, the k^{th} subnetwork will accept a total of $a_A^k + a_B^k$ but not necessarily in the correct proportions from A and B.

To overcome the problem, joint lower bounds are placed on opposite arcs, as in Fig. 61, to ensure the proper amounts are supplied to nodes 1^k and 2^k.

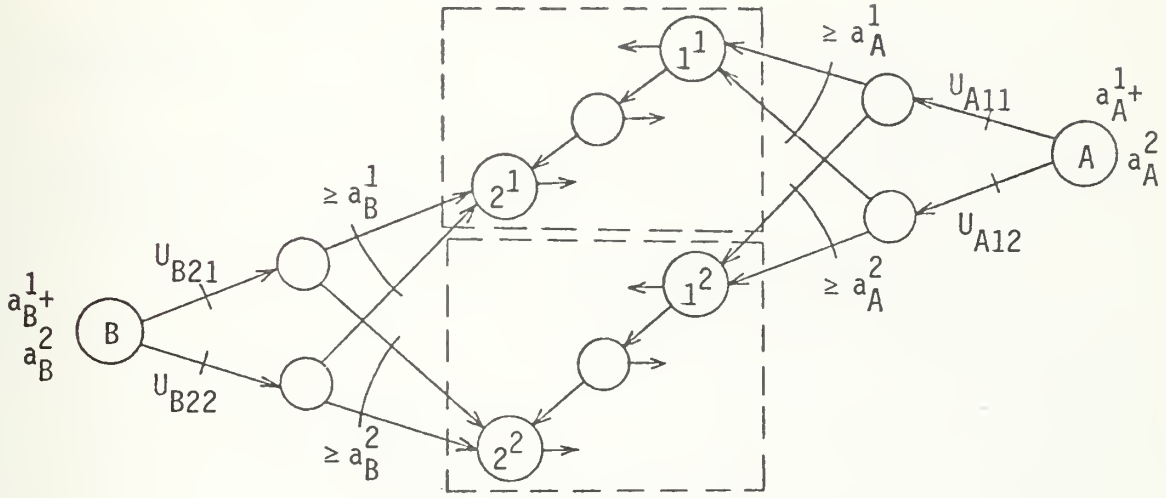
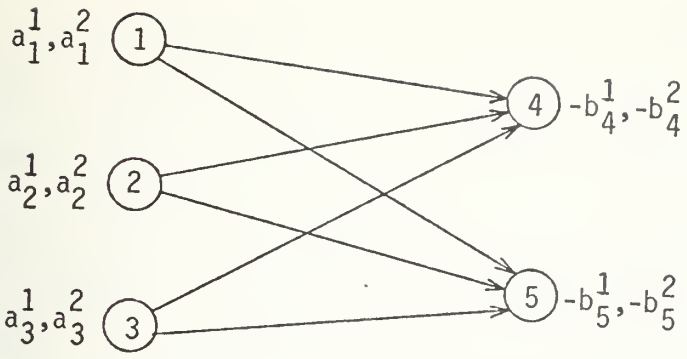


Fig. 61

Transformation VIII can then be applied to remove the lower bounds. Note that this works only because each busy node supplies the rest of the network through only one node. If this were not the case, the lower bounds would be applied across node-disjoint arcs, which violates Condition II in Section II.C. In this problem, either or both A and B could be sinks, and there may be more than two such nodes.

C. RESULT II--THE TWO-SINK (OR SOURCE) TRANSPORTATION PROBLEM

Evans, Jarvis, and Duke [4] have shown that an r-commodity ($r \geq 2$) transportation problem with 2 sinks (or 2 sources) and with all arcs individually capacitated, can be formulated as an uncapacitated single-commodity transportation problem. Figs. 62 and 63 show the transformation for a problem with 3 sources, 2 sinks and 2 commodities.



with $\sum_{k=1}^{r} x_{ij}^k \leq U_{ij}$,

for: $i = 1,2,3$
$j = 4,5$
$r = 2$, in this case.

Fig. 62. The original problem, with 2 commodities and 2 sinks.

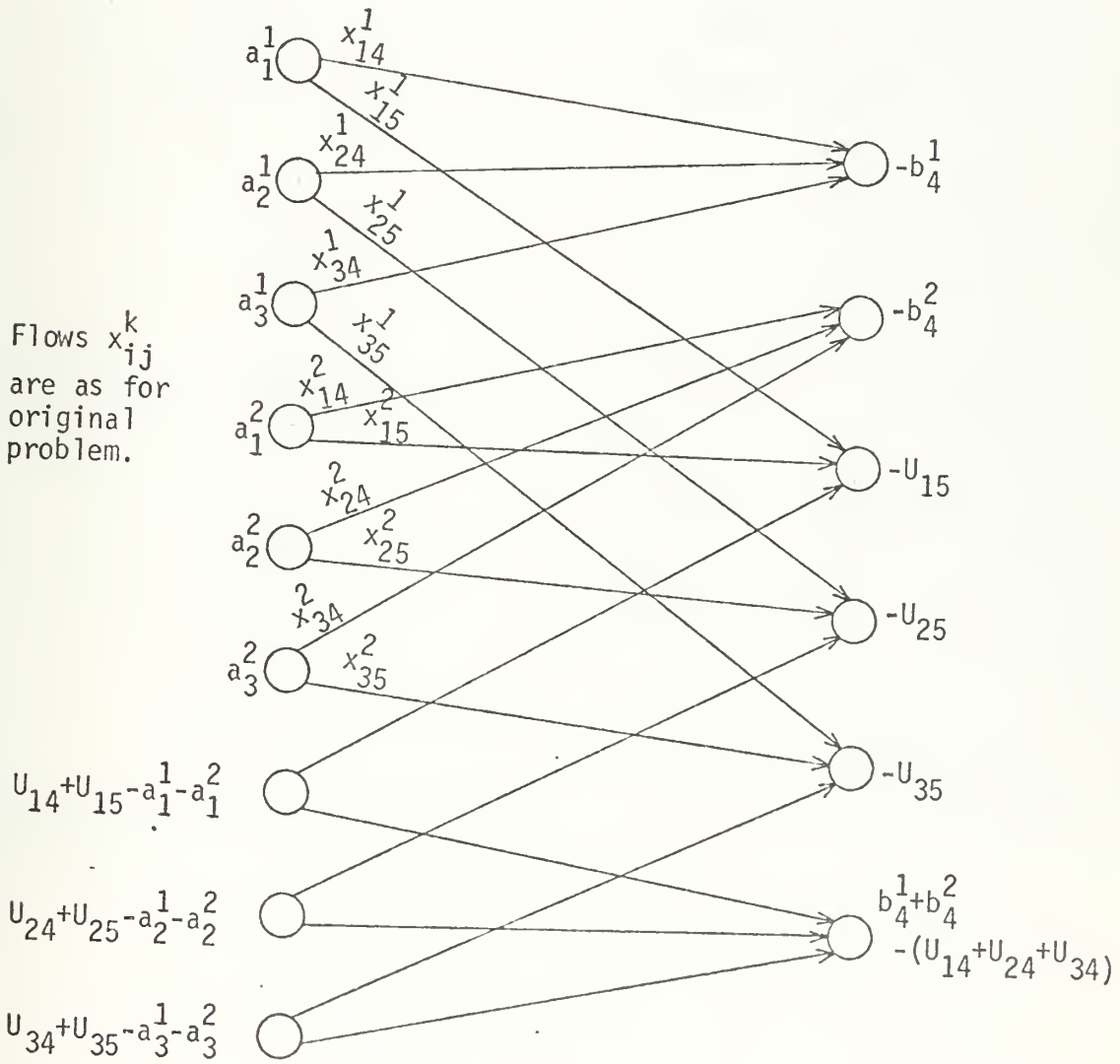
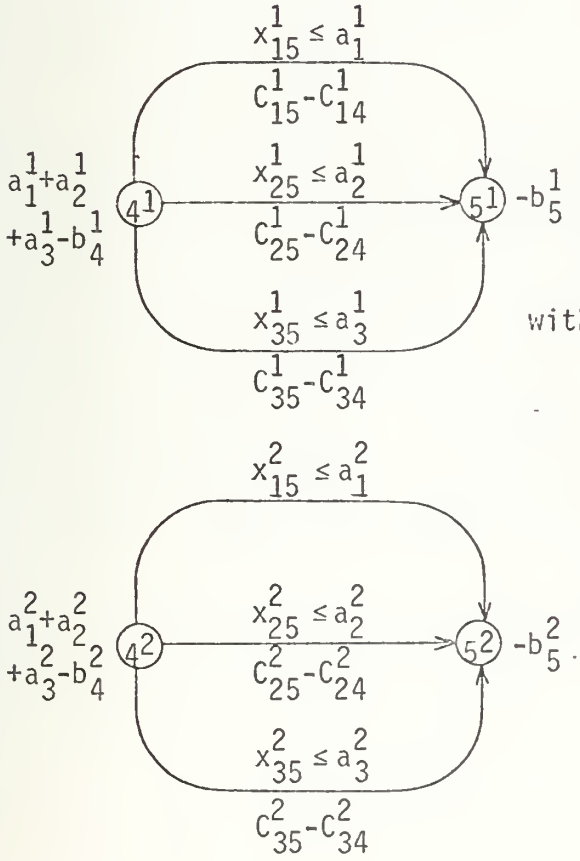


Fig. 63. The re-formulated, single-commodity problem.

In this subsection, the original multicommodity transportation problem is transformed into a capacitated single-commodity transshipment problem, which is considerably simpler than the single-commodity transportation formulation in Fig. 63 above. It may be used in this form with suitable network codes, or it may be further transformed into an uncapacitated transportation problem identical to that in Fig. 63.

Starting with the original problem (Fig. 62), construct r separate subnetworks, one for each commodity. Application of Transformation VII to each and every arc in the subnetworks yields Fig. 64.



with $\sum_{k=1}^{r} a_i^k - U_{i4} \le \sum_{k=1}^{r} x_{i5}^k \le U_{i5}$,

for i = 1,2,3.

$\sum_{k=1}^{r} \sum_{i=1}^{3} a_i^k C_{i4}^k$ is added to the objective function.

Fig. 64

Note that:

1. If r = 1 in the original problem, then the capacity $x_{i5} \leq a_i$ would be made redundant by the capacity $x_{i5} \leq U_{i5}$, since we already have $U_{i5} \leq a_i$ for U_{i5} to be binding. But for $r \geq 2$, <u>both</u> capacities are necessary, $x^k_{i5} \leq a^k_1$ within each subnetwork, and $\sum_{k=1}^{r} x^k_{i5} \leq U_{i5}$ applied across all subnetworks.

2. The lower-bounded capacity $\sum_{k=1}^{r} a^k_i - U_{i4} \leq \sum_{k=1}^{r} x^k_{i5} = \sum_{k=1}^{r} a^k_i - \sum_{k=1}^{r} x^k_{i4}$, and hence is equivalent to $\sum_{k=1}^{r} x^k_{i4} \leq U_{i4}$, which is required in the original problem.

3. The particular requirement for two sinks (or two sources) in the original problem makes transformation to Fig. 64 possible.

Fig. 64 is now equivalent to a single busy node problem with the sink node designated as "busy," with the dependent mass flow conservation constraint equation in each subnetwork. An analogous procedure to the "busy node" proof results in Fig. 65, a single-commodity upper-and-lower bounded transshipment problem. Costs along arcs incident with node B are zero. Subscripts refer to the original problem.

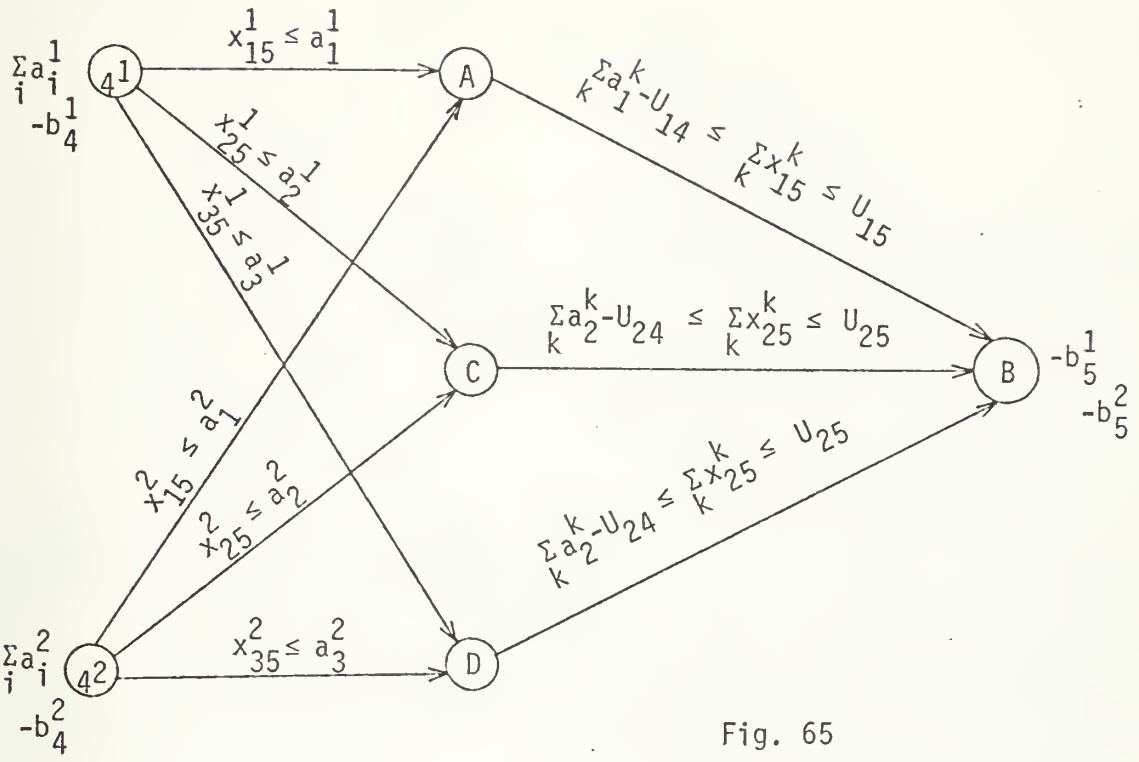


Fig. 65

To get rid of the lower bounds, use Transformation VIII on the appropriate arcs to obtain Fig. 66, a single-commodity capacitated transshipment problem with demand nodes at A, B, C, and D. Since we generally are not interested in the magnitude of resulting flows along arcs incident with node B and since costs are zero along these arcs, Fig. 66 may be directly used in suitable capacitated codes without further post-optimal adjustments to flows along these arcs, or to the objective function (apart from adding $\sum_{k=1}^{r} \sum_{i=1}^{3} a_i^k C_{i4}^k$ as stated in Fig. 64).

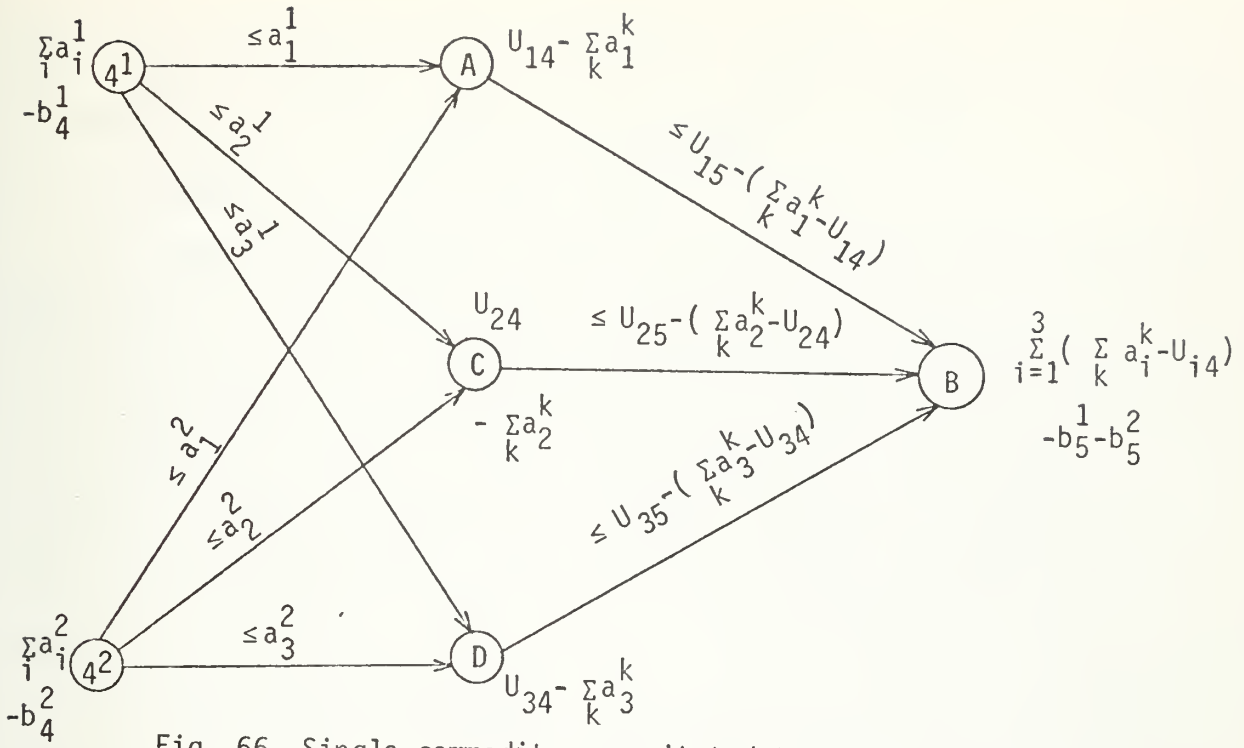


Fig. 66. Single-commodity capacitated transshipment problem.

Finally, to obtain the uncapacitated transportation form, apply Transformation IV to each capacitated arc, to obtain Fig. 67.

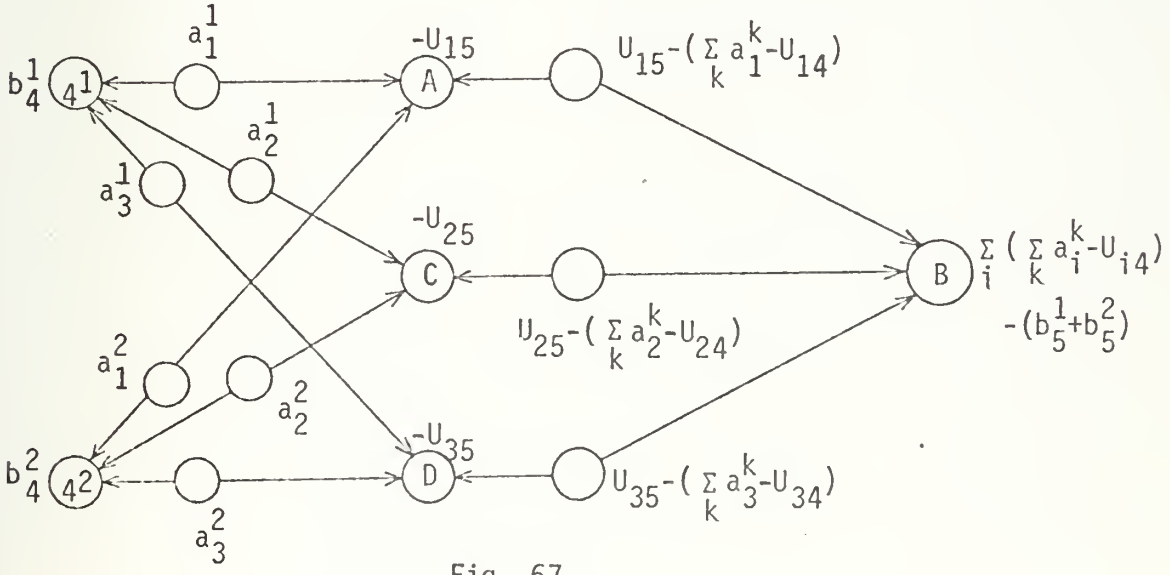


Fig. 67

On re-arranging Fig. 67 (which has a transportation structure) into visually bipartite form, we shall obtain precisely

Fig. 63 with $\sum_i (\sum_k a_i^k - U_{i4}) - (b_5^1 + b_5^2) = \sum_k b_4^k - \sum_i U_{i4}$ at node B, by equality of total demand and supply.

It should be noted again that:

1. The transshipment problem in Fig. 66 has significantly fewer nodes than either the original multicommodity problem, or the original end result in Fig. 63. Since solution times are primarily related to the total number of nodes in a problem, it can be expected that Fig. 66 with a suitable capacitated code should result in considerable savings in time.

2. The result is really a special case of the previously derived extensions to the original "busy node" result described in the previous subsection D.

IV. UNCAPACITATING A TRANSSHIPMENT NETWORK--COMPUTER IMPLEMENTATION

A. GENERAL CONSIDERATIONS

The efficiency of network-based codes lies in carrying out the steps of the SIMPLEX algorithm by manipulation of spanning trees representing successive basic solutions. At the heart of these codes are efficient means of using list structures to store and change a tree structure in computer memory, and to store and manipulate data required by the SIMPLEX algorithm within the tree structure.

The Transformations in Section II change the original network by the addition or deletion of nodes and arcs, and by changes in costs and capacities of arcs. In a network-based code, corresponding changes are necessary to the list structures which define the network and contain the data needed for the SIMPLEX algorithm.

Contemporary efficient network codes are very fast, and so the area of interest lies in large problems where computer time and storage requirements are significant. The most efficient codes store three numbers for each arc, one of which is its capacity. Since most real problems have many more arcs than nodes, there is a considerable saving in computer storage for uncapacitated problems.

Interest is therefore drawn to Transformation IV, which uncapacitates an arc at the cost of adding two additional nodes and arcs. This obviously makes it unattractive for heavily

capacitated problems. Its advantage is likely to lie in problems with only a small percentage of capacitated arcs, where not having to reserve storage for all the arc capacities more than offsets the slight increase in size of the overall network due to application of the Transformation. There may be a penalty, hopefully small, in speed due to the work needed to actually implement the Transformation in the code. However, if a version of the code is written to handle uncapacitated problems, then this time penalty is likely to be more than offset (depending on how lightly capacitated the problem is) by the reduction in time due to simplified solution logic and data manipulation.

B. ASPECTS OF THE IMPLEMENTATION

In this Section, network problems with appropriate characteristics are chosen, and the Transformation implemented by suitably modifying an existing computer code. Particular aspects of the investigation are:

1. Mechanical implementation of the Transformation, i.e., only the original problem is input to the code. Detection of conditions in the problem which require applications of the Transformation, and the consequent applications, are done automatically in the code. This applies also to the extraction of optimal flows and objective function of the original problem from the solution to the uncapacitated problem.

2. Where in the code to apply the Transformation. There are a number of places in the code where the changes of list structures due to the Transformation can be introduced.

3. The effect on the solution time caused by application of the Transformation and the variations in subparagraph 2 above, compared with the case when the Transformation is not applied. It is to be hoped that speed improves, or at least not significantly decreased.

Although the results of the investigation hold only for the particular combination of Transformation, problem type, and code used, there is sufficient similarity with other Transformations or other codes to make the approach used in this investigation to have quite general application.

C. DESCRIPTION OF THE CODE

GNET [10] is a very fast primal network-based code in FORTRAN, designed for large-scale problems by Bradley, Brown, and Graves, which solves the general capacitated transshipment problem (and consequently, the other more specific network models) using the bounded SIMPLEX algorithm.

Its solution technique is built on the structure of basic solutions, which are known to form trees of basic arcs (ignoring orientation) spanning the nodes. The primal pivotal transformations of bases are performed by re-arranging, adding to, and deleting from, these trees structurally, rather than by numerical operations on coefficient matrices.

A detailed description of its workings and its design philosophy is found in Bradley, Brown, and Graves [2]. A brief description of its lists or "arrays" is given below, but this Section is written under the assumption that the reader has [2] available for reference. Note that the GNET

variable names do not necessarily agree with notation used in previous Sections.

1. Arc Data Structure

This group of arrays defines the network structure of the problem. Each arc and its attached information is defined by an entry into every one of these arrays, and they constitute the required input data to GNET. Supplies and demands are input using "dummy" arcs connected to a fictitious super supply and super demand node respectively, but these are discarded after use in the preprocessor subroutine.

M	An integer defining the number of nodes in the problem.
T( )	An arc-length array of tail nodes (i.e., nodes which arcs are directed away from).
H( )	A node-length array of head nodes.
C( )	An arc-length array of unit costs of flows.
CP( )	An arc-length array of arc capacities. In a capacitated problem, arcs may be stored out of the basis with flow equal to capacity, such arcs being marked by a negative sign (-CP( )) for identification.

The reason why H( ) is only of node length is because all arcs with the same head are stored in contiguous space. The arc list may be viewed as sublists of arcs, each sublist belonging implicitly to a particular head node, followed by another sublist belonging to the next larger head node number. The j^{th} entry in H( ), gives the address in the arc list of the first arc with head j.

2. SIMPLEX Data Structure

U( ) The constraints of the dual problem to the primal transshipment problem are of the form:

$C(i,j) - U(j) + U(i) = 0$ for each basic arc (i,j).

The dual variables U( ), also called SIMPLEX multipliers, are stored in a node-length array. Selecting an arc with favorable SIMPLEX cost criteria to enter the current basis involves, in GNET, calculating the dual factor (also called "$Z_j - C_j$" in L.P. terminology) $DF = C(i,j) - U(j) + U(i)$ for each arc in a candidate list, and choosing that arc with the most negative DF.

P( ) If the tree representing a basis is viewed as "hanging" down from a "root" node (the name for such a structure is "arborescence" in graph theory), then a unique "backpath" exists from every node back to the root. Thus, every node (except the root) has a single predecessor node, defined as the first node encountered on this backpath. The minimum information needed to represent a tree is a node-length array P( ) of current predecessors. The orientation of an arc in the tree is indicated by the sign of its predecessor, -P(i) indicating an arc (i,P(i)), and +P(i) indicating (P(i),i). An important use of the predecessor list is in updating the flows in a new basis. If flow is changed in the entering arc (i,j), there is a change of flow only in the arcs of the basis that are in the unique path from i to j in the current tree. This path is identified by iterating the predecessor relationship along the backpaths of i and j until a single node is common to both.

X( ) The current flow X( ) of each basic arc (i,j) is associated with (i.e., indexed by) the deeper node of the pair (see explanation of array D( ) below). For example, flow along basic arc (i,j) is stored in X(i) if j is the predecessor of i.

D( ) For purposes of flow updating, an efficient method of identifying the path from node i to j in the current tree, where (i,j) is the arc to enter the current basis, is to store for each node the number of nodes on the current backpath to the root, called the depth D( ) of the node in the current tree. Depth information indicates which of i,j is deeper and should be iterated using P( ) along its backpath. When both backpaths are at equal depth, the nodes are compared for identity. A match indicates the join node has been reached, otherwise both backpaths are iterated for another comparison. D( ) is also of use in identifying all the nodes belonging to any particular subtree.

IT( ) Suppose a basis arc (i,j) leaves the current basis, where j is the deeper node of the pair i,j. Then, as part of the pivot sequence, only those U( ) associated with nodes in the entire subtree rooted at j are updated from the current basis. Each node in the entire tree is given a current successor node number IT( ), which enables nodes in any subtree to be identified by exhaustively traversing through all its nodes starting from the root. The successor relationship is recursive in that, for any node, its recursive successors are exhaustively traversed before any other nodes of the entire tree, and the rule adopted here is that leftward recursive successors are traversed first. A traversal (also called "pre-order") through any subtree in Fig. 68 clarifies this.

CPX( ) CPX( ) is a node-length array containing the capacity minus the current flow of each current basis arc. Suppose arc (i,j) is to enter the basis. Then the outgoing arc needs to be identified as the arc corresponding to the minimum of:

a. CP(i,j)

b. X( ) along the backpath from i to the join node

c. CPX( ) along the backpath from j to the join node.

This is the "ratio test" of the SIMPLEX algorithm.

Figure 68 below illustrates the main SIMPLEX data array variables in relation to a basis arborescence.

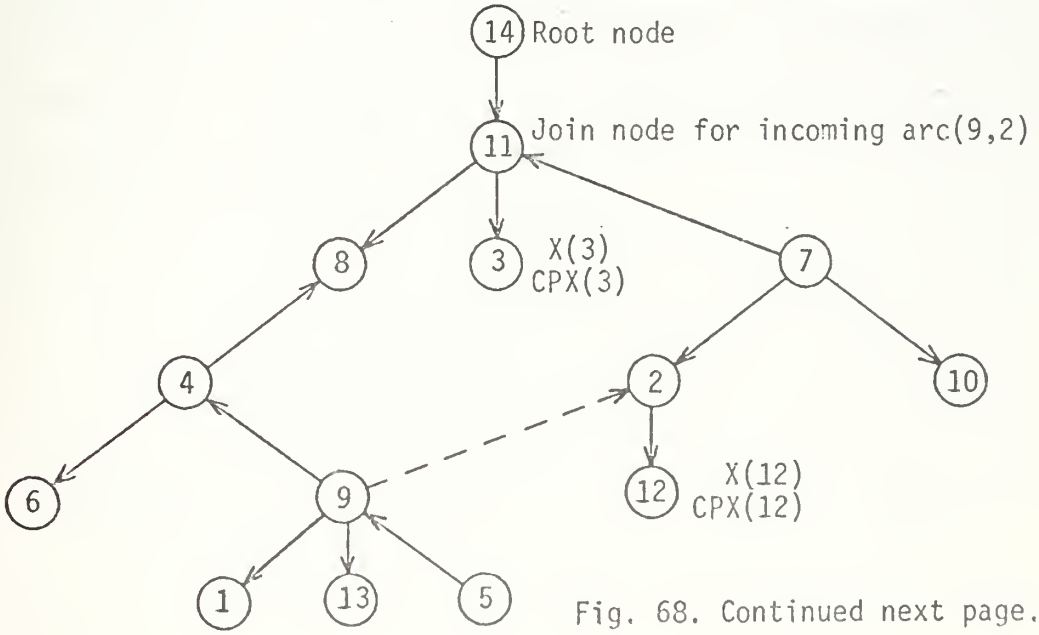


Fig. 68. Continued next page.

Node i:	1	2	3	4	5	6	7	8	9	10	11	12	13	14
P(i) :	9	7	11	-8	-9	4	-11	11	-4	7	14	2	9	--
D(i) :	5	3	2	3	5	4	2	2	4	3	1	4	5	0
IT(i) :	13	12	7	6	3	9	2	4	1	14	8	10	5	11

(9,2) : Incoming arc

CPX(3): Capacity minus flow, for arc (3,11)

X(3) : Flow along arc (3,11)

X(12) : Flow in arc (2,12)

CPX(12): Capacity minus flow, for arc (2,12)

Backpath from node 9: 9, 4, 8, 11, ...

Backpath from node 2: 2, 7, 11, ...

Unique path from 9 to 2 is 9, 4, 8, 11, 7, 2 with 11 the join node.

If (7,11) is outgoing, U( ) for only sub-tree 7, 2, 12, 10 are changed.

GNET introduces an additional node called the root, and sets up an initial arborescence which consists of artificial arcs between each node and the root. The flow in each of these arcs is set equal to the demand or supply of the associated node (arcs are directed to the root from sources and from the root to sinks). The initial arborescence structure appears typically as in Fig. 69.

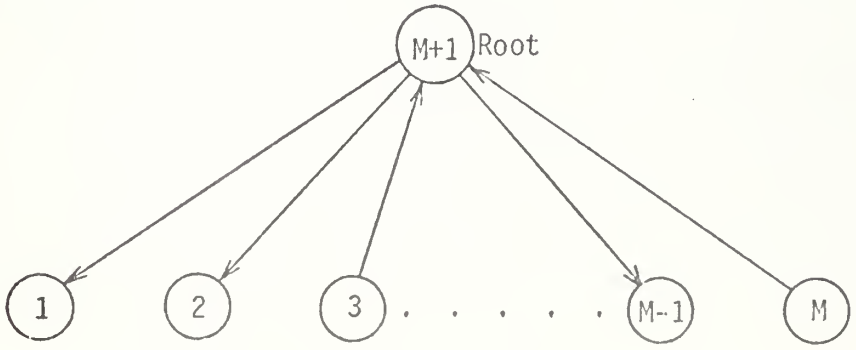


Fig. 69

The rest of the SIMPLEX data structure is set up, with sinks given an arbitrarily large negative value ("Big M") for their U( ). Sources have zero U( ).

3. Preprocessor LNET

GNET includes a preprocessor subroutine LNET which has as its main purposes:

a. To check for certain input errors, and produce statistics on the problem, such as total demand and supply, number of nodes and arcs, ranges of costs and capacities, percent of arcs capacitated (if any), type of problem (capacitated/uncapacitated, transshipment/transportation), etc.

b. To set up the arc data structure in head node sequence as described previously.

c. To attach a X( ) value to each node equal to its supply or demand, which are used by GNET as initial basis flows, as described above. X( ) at sinks are marked with negative signs, which are subsequently removed in GNET after their use for identification purposes.

D. THE TRANSFORMATION CHOSEN FOR IMPLEMENTATION

Transformation IV was chosen for the reasons in subsection A above. Transformation IV uncapacitates an arc, as shown, from Fig. 70 to Fig. 71, by adding two artificial nodes and arcs. Arc (K,j) is chosen to be regarded as containing the original ("real") flow, although (i,K+1) may also be used for this purpose.

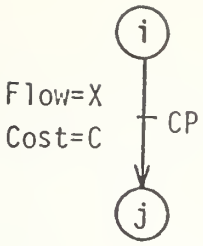


Fig. 70

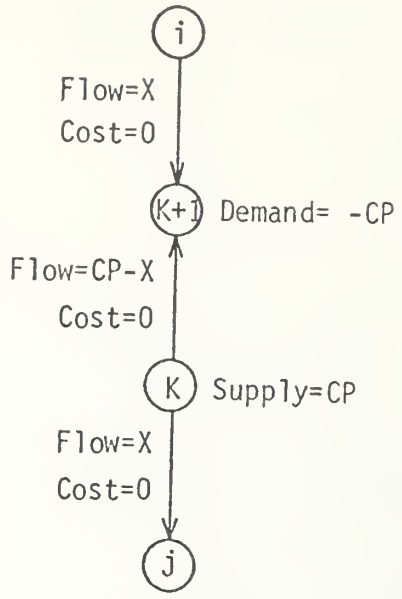


Fig. 71

E. THE CHOSEN PROBLEMS

For reasons given in subsection A above, this Transformation is senn to be most efficient in lightly capacitated problems.

There are some well-known large test problems produced by a program called NETGEN and reported in Ref. 5, which are widely used to test codes. Two of the largest capacitated problems are in fact lightly capacitated and are suitable for this investigation. Their main characteristics are tabled below.

	NETGEN 39	NETGEN 40
Total demand (= total supply)	4,000,000	2,000,000
Number of arcs	15,000	23,000
Number of nodes	5,000	3,000
Percentage of arcs capacitated	0.71%	0.66%
Number of pure sources/sinks	180/700	100/300
Number of transshipment sources/sinks	79/300	49/100
Maximum incoming/outgoing arcs to a node	11/18	20/26
Cost range	1-100	1-100
Capacity range	3,000-91,785	2,000-52,058

Table I. TEST PROBLEM CHARACTERISTICS

These large problems require a lot of computer memory space, so program modifications were tried out on a small problem with 7 nodes before implementation on the large problems.

If the storage space for arc capacities need not be used, this results in respective savings of 15K and 23K single precision numbers, corresponding in the IBM 360 to 60K and 92K bytes respectively, which is substantial.

F. BASIC DATA CHANGES

The Transformation introduces changes to the arc data structure, from Table II to Table III, which correspond to Fig. 70 and Fig. 71 respectively.

Arc List Location Index ℓ		$T(\ell)$	$C(\ell)$	$CP(\ell)$
1				
H(j)				
Arc (i,j) → L	Arcs with head j	T(L)=i	C(L)	CP(L)
H(j+1)-1				
NAR = Total number of arcs				

Table II. ARC DATA STRUCTURE--UNTRANSFORMED PROBLEM

Arc List Location Index ℓ	$T(\ell)$	$C(\ell)$	$CP(\ell)$	
1				
Arc (K,j) → L	T(L)=K	C(L)	MAXCP	
NAR				
Arc (K,K+1) H(K+1)	K	0	MAXCP	↓ Arcs added by Transformations
Arc (i,K+1) H(K+1)+1	i	0	MAXCP	

Table III. ARC DATA STRUCTURE--TRANSFORMED PROBLEM

Two additional entries into the arc list are made for each capacitated arc (i,j) as shown, while the original arc (i,j) becomes (K,j). All arcs are uncapacitated by setting CP( ) = MAXCP, an arbitrarily large number $\geq$ total supply. H(K) = H(K+1), since K is a pure source node, i.e., its arc sublist is vacuous.

G. INSERTION OF DATA CHANGES INTO THE PROGRAM

There were three places in the program where the Transformation was experimentally implemented.

1. Within LNET, where every capacitated arc is transformed. GNET consequently accepts and treats the added arcs and nodes as if they were no different from those in the untransformed problem.

2. As in 1 above, but with modification to GNET so that the initial arborescence is set up with each Transformed arc yielding a structure as in Fig. 71, instead of having all nodes (including the added nodes) at the same depth as normal in GNET (Fig. 69). This is done for all capacitated arcs before pricing-out for the first pivot, and may be termed "pre-hanging."

3. "On-the-fly" in GNET, where the Transformation is applied to a capacitated arc only when it is about to enter a basis.

Since the problem is uncapacitated in all the above cases, there cannot exist non-basic arcs with flow at capacity, i.e., CP( ) is never marked with a negative sign. Two program statements which tested for this condition were removed from

the pricing-out procedure. This was the only modification to solution logic which took into account the uncapacitated nature of the Transformed problem.

A description of each of the three cases follows.

1. Implementation Within LNET

This was the simplest application. After the arc data for all the arcs in the capacitated problem had been structured as normally in LNET, each arc was tested to see if it was capacitated and if so, the data changes in Table III were inserted. Each source node added by a Transformation had positive X( ) value attached, equal to the capacity of the Transformed arc. Sinks had negative X( ) values.

GNET does not differentiate between original and added nodes, and sets up the initial arborescence normally as in Fig. 72.

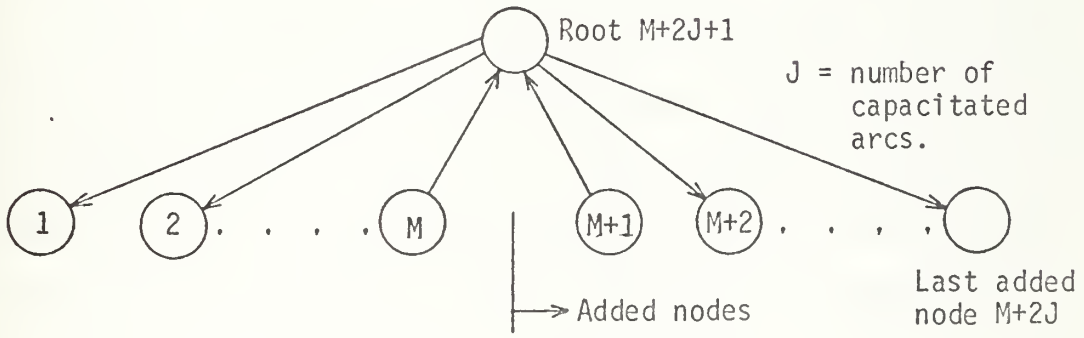


Fig. 72

2. "Pre-hanging"

In this case, changes to data structure occur in LNET exactly as in 1 above. In addition, within the main program, the initial arborescence is set up so that the two nodes k and k+1 which were added in LNET by Transforming capacitated arc (i,j) hang down from the original tail node i. This is done

for every capacitated arc, and amounts to re-structuring nodes M+1 to M+2J in Fig. 72 above to hang below the appropriate original nodes. A typical initial arborescence will look as in Fig. 73 below, where node 2 originally had two capacitated outgoing arcs, for example, and nodes 3 and M are not the tail nodes of any capacitated arcs.

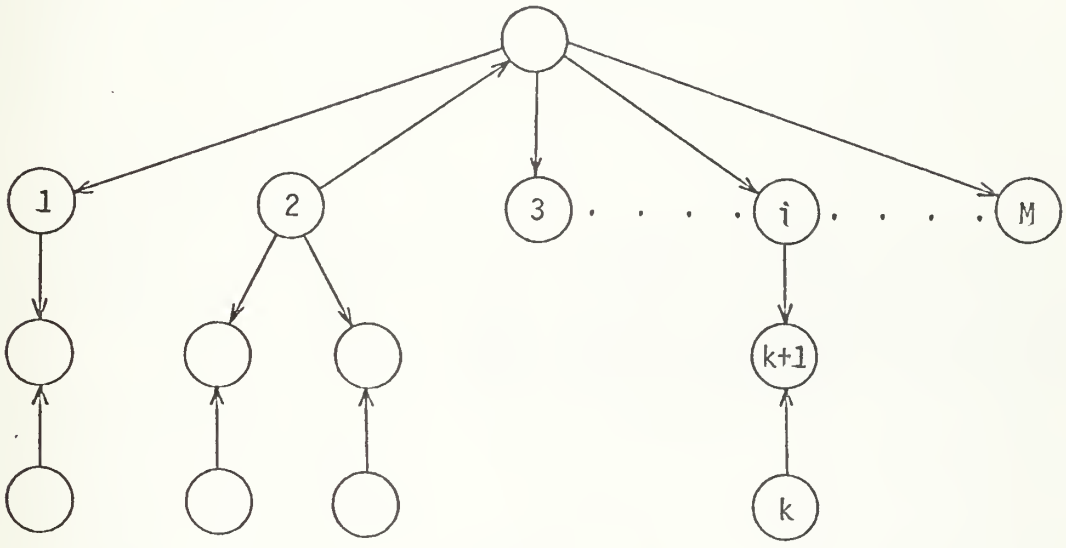


Fig. 73

If the initial arborescence was as in Fig. 72, then before a capacitated arc (i,j) can enter a basis, two prior pivots to bring nodes k, k+1 into the tree must be performed. "Pre-hanging" as in Fig. 73 in effect saves the work needed to perform these two prior pivots, and it was thought that some saving in time due to reduction in total number of pivots to optimality could thereby be saved.

GNET is allowed to set up the normal uniform-depth initial arborescence up to the M^{th} node. All arcs in locations increasing from $\ell = H(M+1)$ are known to have been added in contiguous pairs in LNET, one pair for each arc uncapacitated, as in Table III. Therefore, the following procedure

is adopted for each ℓ, where ℓ begins at ℓ = H(M+1) and increases by steps of 2, (i.e., ℓ, ℓ+2, ℓ+4, ...). The first node k of an added node pair is found from k = T(ℓ). The second added node is immediately obtained as k+1. The original tail node i is found from i = T(ℓ+1). All three nodes needed to pre-hang an added node pair are thus identified, and the structure is defined by setting the following structure variables (relate these to Fig. 73):

P(k)	= - k+1	
P(k+1)	= i	
IT(k)	= IT(i)	If i had more than 1 outgoing capacitated arc, these two statements will cause subsequent node pairs to be pre-hung to the left of preceding pairs.
IT(i)	= k+1	
IT(k+1)	= k	
D(k)	= 3	
D(k+1)	= 2	
IT(M)	= M+2J+1 (the root)	

In addition, the following SIMPLEX data is set:

CPX(k)	= 0	
X(k+1)	= -X(k+1),	[Recall X( ) was set negative by LNET for all sinks.]
CPX(k+1)	= 0	
U(k)	= U(i)	
U(k+1)	= U(i)	

The program then proceeds to the first pivot with the current basis corresponding to the pre-hung arborescence.

3. "Hang-on-the-fly"

This case is the logical extension of the two previous cases. LNET is modified only in order to mark capacitated arcs by a negative T( ). The main program starts off normally with only the original nodes and arcs (including capacitated arcs). The arc (i,j) which is chosen to enter a basis is tested for a negative T( ). If it is a capacitated arc, then:

a. The two additional nodes and arcs are created and corresponding changes made in the arc data structure as in case 1 above.

b. The two additional nodes, k, k+1, are pre-hung into the arborescence as in case 2, from node i. The arc to enter the basis is now (k,j).

c. Appropriate SIMPLEX data is attached to the added nodes.

The above procedure is essentially a simultaneous combination of cases 1 and 2, performed between the pricing-out and pivot sequences in the main program. The motivation behind hang-on-the-fly is to minimize the number of added nodes in bases by creating them only when a capacitated arc is about to enter. This should reduce the overall amount of work done in arborescence traversing, and updating the arborescence and SIMPLEX data in the pivot sequence. The size of the network and hence the basis arborescence grows as the number of pivots increases, since nodes added by transforming entering capacitated arcs remain as part of the network.

This is in contrast to cases 1 and 2, where the network is expanded to full (constant) size at the outset.

H. RESULTS AND DISCUSSION

The effect of the Transformation on the speed of solution in the various cases examined is shown below.

	NETGEN 39		NETGEN 40	
	Time (seconds)	Number of Pivots	Time (seconds)	Number of Pivots
Unmodified GNET (capacitated problem)	116.3	9553	67.0	6409
Transformation within LNET	116.3	9615	70.8	7511
Pre-hanging	118.0	9689	67.2	7052
Hang-on-the-fly	113.8	9652	64.7	6733

Table IV. COMPUTER IMPLEMENTATION RESULTS

GNET is written in standard ANSI FORTRAN. All computer runs were on the Naval Postgraduate School's IBM 360/67. The solution times given below are actual execution times, not clock times.

It is clear that the results are quite uniform across all cases in speed of solution or number of pivots in the above cases, for each problem. It is possible to discern a slight improvement by hanging-on-the-fly compared to the other cases.

Thus, speed performance is maintained even with the Transformation implemented. This is significant in two respects:

1. If a version of GNET is developed to specifically solve uncapacitated problems, solution speed should be at

least maintained while there would be a considerable saving in storage presently required to store arc capacities. This would enable still larger or more heavily capacitated problems to be accommodated.

2. Similarly, an uncapacitated GNET version would mean deletion of the solution logic and data handling designed specifically for the capacitated problem. This should give a speed improvement compared to the same problem solved in the current GNET.

Overall, therefore, an uncapacitated GNET should solve larger or more heavily capacitated problems for little speed loss, or the same (lightly capacitated) problem at a greater speed.

Of course, if the code available can only solve the uncapacitated transportation problem, then use of this Transformation becomes a matter of necessity rather than efficiency.

For progressively more heavily capacitated problems, the saving in storage requirements due to deletion of CP( ) in an uncapacitated version of GNET would be increasingly offset by the storage needed by the expanded size of the problems. Each application of the Transformation adds two more nodes and arcs. In the extreme case of a 100% capacitated problem, the number of arcs will expand by a factor of three, and this is clearly well beyond the point where application of the Transformation would be considered worthwhile.

LIST OF REFERENCES

1. Bradley, G. H., "Survey of Deterministic Networks," AIIE Transactions, 7, 3, p.222-234, September 1975.

2. Bradley, G. H., Brown, G. G., Graves, G. W., "A Comparison of Data Structures for Primal Network Codes," (in preparation).

3. Dantzig, G., Linear Programming and Extensions, Princeton University Press, 1963.

4. Evans, J. R., Jarvis, J. J., Duke, R. A., Matroids, Unimodularity, and the Multicommodity Transportation Problem, presented at the Joint ORSA/TIMS National Meeting, Chicago, Illinois, 1975.

5. Klingman, D., Napier, A., Stutz, J., "NETGEN--A Program for Generating Large Scale (Un)Capacitated Assignment, Transportation, and Minimum Cost Flow Network Problems," Management Science, 20, 5, p.814-822, 1974.

6. Koopmans, T. C., "Optimum Utilization of the Transportation System," Proceedings of the International Statistical Conferences, Washington, D. C., 1947.

7. Rebman, K. R., "Total Unimodularity and the Transportation Problem: A Generalization," Linear Algebra and Its Applications, 8, p.11-24, 1974.

8. Waggener, H. A., Suzuki, G., "Bid evaluation for Procurement of Aviation Fuel at DSFC: A Case History," Naval Research Logistics Quarterly, 14, March 1967.

9. Wagner, H. M., Principles of Operations Research, 2nd ed., Prentice-Hall, 1975.

10. GNET, a capacitated transshipment network code, Copyright 1975. For availability, contact authors of Ref. 2 above.

INITIAL DISTRIBUTION LIST

		No. Copies
1.	Defence Documentation Center Cameron Station Alexandria, Virginia 22314	2
2.	Library, Code 0212 Naval Postgraduate School Monterey, California 93940	2
3.	Department Chairman, Code 55 Department of Operations Research and Administrative Sciences Naval Postgraduate School Monterey, California 93940	2
4.	Assoc Professor Gordon H. Bradley, Code 55Bz Department of Operations Research and Administrative Sciences Naval Postgraduate School Monterey, California 93940	10
5.	Asst Professor Gerald G. Brown, Code 55Zr Department of Operations Research and Administrative Sciences Naval Postgraduate School Monterey, California 93940	2
6.	MAJ Yue Pui Cheong, Singapore Armed Forces Ministry of Defence Tanglin Barracks Singapore 10 Republic of Singapore	2
7.	Assoc Professor James K. Hartman, Code 55Hh Assoc Professor Gilbert T. Howard, Code 55Hk Assoc Professor Alan W. McMasters, Code 55Mg Department of Operations Research and Administrative Sciences Naval Postgraduate School Monterey, California 93940	1 1 1
8.	MAJ Calvin Anderson, USA Concepts Analysis Agency 8120 Woodmont Avenue Bethesda, Maryland 20014	1

9.	Professor George B. Dantzig Department of Operations Research Stanford University Stanford, California 94305	1
10.	Dr. Neal Glassman, Office of Naval Research Arlington, Virginia 22217	1
11.	Professor Glenn Graves Graduate School of Business Administration University of California Los Angeles, California 90024	1
12.	MAJ Michael Hanley, USMC NMCSSC - DCA Room BE-685, Pentagon Washington, D.C., 20301	1
13.	Professor John Jarvis School of Industrial and Systems Engineering Georgia Institute of Technology Atlanta, Georgia 30332	1